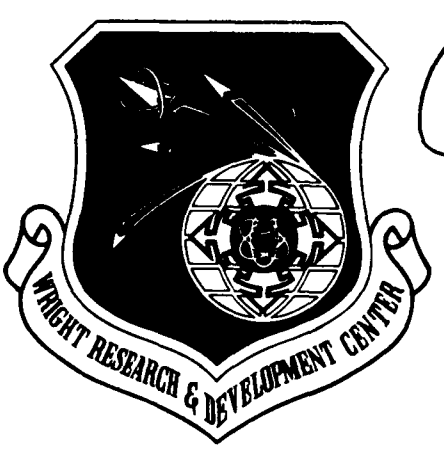
WRDC-TR-90-8007
Volume V
Part 9
Section 4 of 5



INTEGRATED INFORMATION SUPPORT SYSTEM (IISS)
Volume V - Common Data Model Subsystem
Part 9 - Neutral Data Manipulation Language (NDML) Precompiler Development Specification
Section 4 of 5

J. Althoff, M. Apicella

Control Data Corporation
Integration Technology Services
2970 Presidential Drive
Fairborn, OH 45324-6209

September 1990

Final Report for Period 1 April 1987 - 31 December 1990

MANUFACTURING TECHNOLOGY DIRECTORATE
WRIGHT RESEARCH AND DEVELOPMENT CENTER
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6533

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever, regardless whether or not the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data. It should not, therefore, be construed or implied by any person, persons, or organization that the Government is licensing or conveying any rights or permission to manufacture, use, or market any patented invention that may in any way be related thereto.

This technical report has been reviewed and is approved for publication.

This report is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations

DAVID L. JUDSON, Project Manager
WRDC/MTI
Wright-Patterson AFB, OH 45433-6533

25 July 91
DATE

FOR THE COMMANDER:

BRUCE A. RASMUSSEN, Chief
WRDC/MTI
Wright-Patterson AFB, OH 45433-6533

25 July 91
DATE

If your address has changed, if you wish to be removed form our mailing list, or if the addressee is no longer employed by your organization please notify WRDC/MTI, Wright-Patterson Air Force Base, OH 45433-6533 to help us maintain a current mailing list.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

REPORT DOCUMENTATION PAGE

FORM APPROVED
OMB NO. 0704-0188

1. AGENCY USE ONLY *(Leave Blank)*

2. REPORT DATE
September 1990

3. REPORT TYPE AND DATES COVERED
Final Technical Report
1Apr87 - 31Dec90

4. TITLE AND SUBTITLE
INTEGRATED INFORMATION SUPPORT SYSTEM (IISS)
Volume V - Common Data Model Subsystem
Part 9 - Neutral Data Manipulation Language (NDML) Precompiler Development Specification
Section 4 of 5

5. FUNDING NUMBERS
Contract No.: F33600-87-C-0464
PE: 78011F
Proj. No.: 595600
Task No.: P95600
WU: 20950607

6. AUTHOR(S)
J. Althoff, M. Apicella

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Controld Data Corporation
Integration Technology Services
2970 Presidential Drive
Fairborn, OH 45324-6209

8. PERFORMING ORGANIZATION REPORT NUMBER
DS 620341200

9. SPONSORING MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Manufacturing Technology Directorate (WRDC/MTI)
Wright-Patterson AFB, OH 45433-6533

10. SPONSORING/MONITORING AGENCY REP NUMBER
WRDC-TR-90-8007, Vol. V, Part 9
Section 4 of 5

11. SUPPLEMENTARY NOTES
WRDC/MTI Project Priority 6203

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for Public Release; Distribution is Unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT

This development Specification (DS) describes the functions, performance, environment, interfaces, and design requirements for the Neutral Data Manipulation Language (NDML) Precompiler. The NDML Precompiler is a component of the Common Data Model Processor (CDMP) and it is used to generate various programs (e.g., request processor or RP, RP drivers, CS-ES transformers, and local subroutine callers) tailored to satisfy the NDML requests in a specific application program.

This report is divided into five (5) sections.

14. SUBJECT TERMS

15. NUMBER OF PAGES
885

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASS OF THIS PAGE.	19. SECURITY CLASS OF ABSTRACT	20. LIMITATION ABSTRACT
SAR	SAR	SAR	SAR

SECTION 24

FUNCTION PRE9.3 GENERATE CODASYL REQUEST PROCESSOR

This program (CDQPC) is a collection of runtime modules whose purpose is to generate the COBOL code necessary to satisfy a NDML subtransaction request against a CODASYL database.

Two reference tables are contained in this section. Table 24-1 is a cross reference of the code that is generated from generic DML commands. Table 24-2 contains a listing of all code macros used by the CODASYL Request Processor Generator.

24.1 Inputs

Inputs to the CODASYL Request Processor are outputs from the following programs:

PRE5 - Decompose CS NDML
PRE6 - Select IS Access Path
PRE7 - Transform IS Access Path/Generic DML

1. Conceptual Schema Formats

The Result Field Table contains the CS descriptions of the data fields processed by the NDML transaction.

The CS-QUALIFY-LIST contains the CS description of the variables and constants which are passed to the RP at runtime.

The CS-ACTION-LIST contains the CS descriptions of the data fields requested by the NDML transaction.

2. Generic DML Commands

The GC-TABLE contains the generic DML commands to process the NDML request.

The Record Key Table contains the record keys for TOTAL access path being processed.

3. Access Path Information

The Access Path Selector Tables contain transaction data for the access path being processed.

4. Internal Schema Formats

The IS-ACTION-LIST contains the IS descriptions of the data fields processed by the NDML transaction.

The IS-QUALIFY-LIST contains the IS description of the variables and constants which are passed to the RP at runtime.

5. Database Information

Name of database being accessed (DB-ID)
Schema name of the database being accessed (SCHEMA-NAME)
Subschema name of the database being accessed (SUBSCHEMA-NAME)
DBMS of the database being accessed (DMBS-NAME)
Macro library name of database being accessed (LIB-NAME)
System database location (DB-LOCATION)
Numeric null value for database (NUMERIC-NULL-VALUE)
Character null value for database (CHARACTER-NULL-VALUE)

6. Request Processor Information

Subtransaction identification number (SUB-ID)
Request processor program name (RP-NAME)
Current host (CURRENT-HOST)
Target host (TARGET-HOST)
Error file name (ERROR-FILE)

24.2 Processing

The CODASYL Request Processor generator (CDQPC) uses two sequential work files to allow WORKING-STORAGE and PROCEDURE DIVISION statements to be constructed simultaneously. At the end of processing, the two files are combined to produce the working RP. Code that is common to all RPs is generated utilizing the macro replacement utility CDMACR. Table 14-1 contains a listing of the code contained within these macros. Code specific to the CODASYL RP is derived from information in various tables, variables and CDM meta data. Errors are reported using one of the three error reporting routines, depending on the type of error. User errors are reported by `RPTERR', system errors by `ERRPRO'. User errors consist of CDM meta data errors. All other types of errors, such as table overflows and DBMS errors, are considered system errors.

1. Initialize the Request Processor generator internal tables and variables.

   Call routine GENFIL to establish the names of the two work files to be used during the generation of the Request Processor (FILE-NAME1, FILE-NAME2).

2. Generate the start of the COBOL Program.

   Generate:

   ```
   IDENTIFICATION DIVISION
   PROGRAM-ID
   ENVIRONMENT DIVISION
   INPUT-OUTPUT SECTION
   FILE-CONTROL
   DATA DIVISION
   ```

   using the Macro Generator "CDMACR" with QPG01 as a parameter.

3. Generate the FD data definitions for the results file of retrieved data and working storage data definitions for the retrieved data by calling "CDRFT" with work file name, subtransaction identification, and the result field table as inputs.

   Call "CDRFT" with the following:

```
FILE-NAME1
SUB-ID
RFT
```

CDRFT will generate:

```
01   RESULTS-REC.
     03    RES-NULL-nn        PIC 9.
     03    RES-nn             PIC type(size)[V9(nd)].
WORKING-STORAGE SECTION.
01   CS-VAR-nn                PIC type(size)[V9(nd)].
```

4. Generate the schema and sub-schema statements of the request processor using information from input parameters.

5. Generate the code in WORKING STORAGE for the CODASYL error status variables.

   5.1 Call "CDMACR" using QPG02 as the macro name.

6. This step has been replaced with step 10.2.

7. This step has been replaced with step 10.3.

8. Generate internal schema data definitions required for search/update values, retrieved data fields and qualification data fields.

   8.1 Generate internal schema data definitions required for the transformation of the runtime search/update values from conceptual to internal format which will be input by the user at runtime.

   Call "CDPRM" with the following:

```
FILE-NAME1
SUB-ID
IS-QUALIFY-LIST
COMPLEX-MAPPING-ALG-TABLE
```

   CDPRM will generate:

```
01  ISQL-VAR-nn           PIC type(size)
                               [V9(nd)].
01  ISQL-mod-name-mm-nn   PIC type(size)
[V9(nd)].
```

8.2 Generate the internal schema data definitions for the retrieved data fields and those that will be used for retrieval qualifications.

Call "CDQDF" with the following:

```
FILE-NAME1
SUB-ID
IS-QUALIFY-LIST
COMPLEX-MAPPING-ALG-TABLE
```

CDQDF will generate:

```
   01  IS-VAR-nn              PIC type(size)
[V9(nd)].
   01  IS-mod-name-mm-nn      PIC type(size)
[V9(nd)].
```

8.3 "CDRDF"

9. Generate the complete definition of the record to be returned by the DBMS.

For each unique record name (IS-RTID) found in the retrieve list (IS-ACTION-LIST):

9.1 Generate the 01 level entry for the record definition:

```
   01   R-rtno.
```

where

```
   rtno = IS-RTNO
```

9.2 Call "CDGENRT" with the following:

```
IS-DBID
IS-RTID
IS-RTNO
FILE-NAME1
MODULE-STATUS
```

10. Generate the remainder of the WORKING STORAGE section and the linkage section for the request processor.

10.1 Generate the data definitions for complex mapping algorithm parameters.

Call "CDCMPRM" with the following:

```
FILE-NAME1
COMPLEX-MAPPING-ALG-TABLE
SUB-ID
MODULE-STATUS
```

CDCMPRM will generate:

```
01  PARM-mod-name-mm-nn      PIC type(size)
                                [V9(nd)].
```

10.2 Generate the conceptual schema data definitions for the runtime update/search values and the start of the Linkage Section for the request processor program.

Call "CDMSG" with the following:

```
FILE-NAME1
SUB-ID
CS-QUALIFY-LIST
CS-ACTION-LIST
IS-QUALIFY-LIST
IS-ACTION-LIST
```

CDMSG will generate:

```
01  CS-VAR-nn        PIC type(size)[V9(nd)].
01  CSQ-VAR-nn       PIC type(size)[V9(nd)].
LINKAGE SECTION.
01  MESSAGE-BODY-IN.
    03  CASE-NO      PIC XXX.
    03  SUBID        PIC 999.
    03  MSG-VAR-nn   PIC type(size)[V9(nd)].
    03  MSGI-VAR-nn  PIC type(size)[V9(nd)].
```

10.3 Generate data definition for the output parameters of the request processor program.

Call "CDMACR" utility with QPG03 as the macro name.

11. Generate the PROCEDURE DIVISION.

11.1 Call the macro expander "CDMACR" using QPG05 as the macro name.

12. This step has been removed.

13. This step has been removed.

14. This step has been removed.

15. Generate the CODASYL Request Processor Data retrieval/update code.

    15.1 Process each command in the GC-TABLE and generate the appropriate code as depicted in Table 14-2.

    15.2 If the GC command is one that will produce an "IF" statement (IF, IN1, IN2, IRF, IRN), then special processing must be performed in order to handle the possibility of nested IFs in the generated request processor.

        15.2.1 Determine if the next GC command is an `EIF'. If so, then set the GC-INDEX up by one and exit from processing the current command.

        15.2.2 Stack the GC-INDEX of the current command on the IF stack

            IF-STACK-PTR = GC-INDEX
            IF-STACK-USED = IF-STACK-USED+1

        15.2.3 Set the IF command flag to "on" status

            IF-COMMAND-FLAG = 1

        15.2.4 Continue processing with the next GC command.

    15.3 If the GC command is one that will not produce an "IF" statement and is not an "EIF" (ENDIF) command, then determine if this command is within an IF END-IF structure. Determine if the IF-COMMAND-FLAG is set.

        15.3.1 If the IF-COMMAND-FLAG is set

            15.3.1.1 Generate an IF statement for each of the IF commands on the IF stack.

15.3.1.2 Set the IF-COMMAND-FLAG to zero.

15.3.1.3 Generate the code of the current command being processed.

15.3.2 If the IF-COMMAND-FLAG is not set

15.3.2.1 Generate the code for the current command being processed.

16. For each entry in the GC-TABLE, determine the GC command and perform the appropriate section of code as detailed in steps 17-84. After all GC commands have been processed, continue processing at step 85.

17. This section will generate the code for the "END IF".

17.1 Generate a period (.) in the PROCEDURE DIVISION of the RP.

17.2 Search forward in the GC-TABLE for the next non-EIF command or end the table condition.

17.3 For each EIF command found in the above search, `pop' the IF stack one level.

17.4 When the next new EIF command is found, set the GC-INDEX down by one.

18. This section will generate the code for the ELP command (END LOOP).

18.1 Generate the following:

GO TO LOOP-nn.

EXIT-nn.

18.2 Generate GO TO LOOP-nn. If GC-SPEC-PTR equals zero, go to Step 18.3.

Otherwise set

nn = GC-SPEC-PTR

18.3 Generate

EXIT-nn.

Set

nn = GC-SPEC-PTR.

19. This section will generate the SE" COBOL statement. Generate the followi.

ELSE

20. This step has been replaced by steps 84-85.

21. This section of code will generate the FFR FIND FIRST RECORD statement using a CALC key.

21.1 Generate a MOVE statement to set up the key prior to the call as follows:

MOVE ISQL-VAR-NN to RK-DFID of RK-RTID.

where NN is the value of the index and RK-DFID and RK-RTID are the names of the key field and record type respectively passed to the program in structure AP-INFO-TABLE

21.2 Generate the following:

FIND FIRST recid.

where recid is the value in RK-RTID passed to PRE9 in structure AP-INFO-TABLE

22. This section of code will generate the FIND FIRST REC WITHIN AREA statement. (FFA)

22.1 Generate the text.

FIND FIRST recid WITHIN areaid.

where recid is the value held in RAS-RTID and areaid is the value in RAS-AREAID of the structure AP-INFO-TABLE

22.2 Generate code to test the return status

23. This section of code will generate the FIND RECORD WITHIN SET statement. (FFM)

    23.1 Generate the following statement:

         FIND FIRST recid WITHIN setname.

         where recid is the value in SS-RTID and setname is the value in SS-SETID of the structure AP-INFO-TABLE

    23.2 Generate code to test the return status

24. This section of code will generate the FIND NEXT RECORD WITHIN AREA statement. (FNA)

    24.1 Generate the following statement:

         FIND NEXT recid WITHIN areaid.

         where recid and areaid are the values passed to PRE9 in the structure AP-INFO-TABLE

    24.2 Generate the code to test the return status.

25. This section of code will generate the FIND NEXT RECORD WITHIN SET statement. (FNM)

    25.1 Generate the following statement:

         FIND NEXT recname WITHIN setname.

         where recname and setname are the values in SS-RTID AND SS-SETID in the structure AP-INFO-TABLE passed to PRE9

    25.2 Generate the code to test the return status.

26. This section will generate the code to generate a FIND next record in CALC mode. (FNR)

    26.1 Generate the following:

         FIND NEXT recname.

         where recname is the value in RK-RTID of the structure AP-INFO-TABLE which was passed to PRE9.

26.2 Generate the code to test the return status.

27. This section will generate the FIND OWNER IN SET statement. (FOW)

27.1 Generate the following:

FIND OWNER WITHIN setname.

where setname is the value in SS-SETID of the structure AP-INFO-TABLE

27.2 Generate the code to test the IDMS return status.

28. This section will generate the code required for an IF statement. (IN3) The IF statement is of the following form:

IF NOT (dbitem OP variable)

28.1 Generate

IF NOT (dbitem OP

where dbitem can take the following forms:

If the value in the record id (RS3-RTID) is blank then this is AUC settype value so generate

"RS3-DFID"

else generate

RS3-DFID OF RS3-RTID

where RS3-DFID is the data field name and RS3-RTID is the record name.

28.2 Now generate the variable as follows:

If RS3-SIDE = "L", then generate

ISQL-VAR-NN)

else generate

ISQR-VAR-NN)

29. Generate code to determine if a retrieved data field contains a null value and establish the contents of the variable and its null flag declared in the working storage section. Generate code to perform numeric checks on retrieved data before moving it to a local variable to avoid abnormal end of processing. (GIO)

Generate the following if FG1-TYPE = "C"

```
If  D-dfno = null-value
    MOVE 1 TO ISQside-NULL-nn
    MOVE ZEROES TO ISQside-VAR-nn
If  D-dfno NOT = null-value
    MOVE 0 TO ISQside-NULL-nn
    MOVE D-dfno TO ISQ-side-VAR-nn
```

Generate the following if FG1-TYPE NOT = "C"

```
IF D-dfno NOT NUMERIC
  STRING "DATA CONTAINED IN DATA FIELD" FG1-DFID
     (GC-SPEC-PTR (GC-INDEX))
     "FOR RECORD"
     FGI-RTID (GC-SPEC-PTR (GC-INDEX))
     "IS NON NUMERIC"
  DELIMITED BY SIZE
  INTO MESG-DESC
  PERFORM PROCESS-ERROR
  MOVE 1 TO ISQside-NULL-nn
  MOVE ZEROES TO ISQside-VAR-nn
ELSE
  IF D-dfno = null-value
     MOVE 1 TO  ISQside-NULL-nn
     MOVE ZEROES TO ISQside-VAR-nn
  ELSE
     MOVE 0 TO ISQside-NULL-nn
     MOVE D-dfno TO RES-nn
```

where:

```
dfno       = FG1-DFNO    (GC-SPEC-PTR(GC-INDEX))
side       = FG1-SIDE    (GC-SPEC-PTR(GC-INDEX))
nn         = FG1-ISQ-PTR (GC-SPEC-PTR(GC-INDEX))
null-value = CHARACTER-NULL-VALUE from input
       parameter if:
             FG1-TYPE (GC-SPEC-PTR(GC-INDEX)) = "C"
             = NUMERIC-NULL-VALUE from input
```

```
                    parameter if:
                  FG1-TYPE (GC-SPEC-PTR (GC-INDEX)) NOT
          = "C"
```

30. Generate code for an IF condition check of a value and a variable holding a retrieved data field. (IF)

Generate the following procedure division statement:

```
   IF (value op variable nn)
```

where:

```
value    = IT2-VALUE  (GC-SPEC-PTR(GC-INDEX))
op       = IT2-OP     (GC-SPEC-PTR(GC-INDEX)
nn       = IT2-ISQ-PTR(GC-SPEC-PTR(GC-INDEX))
variable = "IS-VAR" if
            IT2-ISQ-PTR(GC-SPEC-PTR(GC-INDEX)) = 0
         = "ISQL-VAR" if
            IT2-ISQ-PTR(GC-SPEC-PTR(GC-INDEX)) NOT = 0
```

31. Generate code for an IF NOT condition of two retrieved data fields. (IN1)

Generate the following procedure division statements:

```
IF NOT (D-dfno1 = null-value1 AND
        D-dnfo2 = null-value2) AND
   NOT (D-dfno1 op D-dfno2)
```

where

```
null-value1 = CHARACTER-NULL-VALUE from input
parameter if:
              RS1-TYPE1 (GC-SPEC-PTR(GC-INDEX)) = "C"

              NUMERIC-NULL-VALUE from input
              parameter if:
              RS1-TYPE1 (GC-SPEC-PTR(GC-INDEX)) NOT =
              "C"
null-value2 = CHARACTER-NULL-VALUE from input
              parameter if:
                 RS1-TYPE2 (GC-SPEC-PTR(GC-INDEX)) = "C"
            = NUMERIC-NULL-VALUE from input
              parameter if:
                 RS1-TYPE2 (GC-SPEC-PTR(GC-INDEX)) NOT = "C"
dfno1       = RS1-DFNOL (GC-SPEC-PTR(GC-INDEX))
op          = RS1-OP    (GC-SPEC-PTR(GC-INDEX))
dfno2       = RS1-DFNOR (GC-SPEC-PTR(GC-INDEX))
```

32. Generate code for an IF NOT condition of a data field and a variable holding a retrieved data field. (IN2)

Generate one of the following procedure division statements.

If RS4-SIDE (GC-SPEC-PTR(GC-INDEX)) = "L" generate:

```
IF NOT (D-dfno = null-value)
AND NOT (D-dfno op ISQL-VAR-nn)
```

IF RS4-SIDE (GC-SPEC-PTR(GC-INDEX)) = "R", generate:

```
IF NOT (D-dfno = null-value)
AND NOT (D-dfno op ISQR-VAL-nn)
```

where:

```
dfno       = RS4-DFNO    (GC-SPEC-PTR(GC-INDEX))
op         = RS4-OP      (GC-SPEC-PTR(GC-INDEX))
nn         = RS4-ISQ-PTR (GC-SPEC-PTR(GC-INDEX))
null-value = CHARACTER-NULL-VALUE from input
parameter if:
               RS4-TYPE (GC-SPEC-PTR(GC-INDEX)) = "C"
           = NUMERIC-NULL-VALUE from input
parameter if:
          RS4-TYPE (GC-SPEC-PTR(GC-INDEX)) NOT = "C"
```

33. Generate the following for an IRF command:

IF ((OK) AND NOT (EOS or EOR or EOC or EOO))

34. Generate the following for an IRN command:

IF ((EOS or EOA or EOC or EOO) AND (OK))

35. This section will generate a label and a location tracking move. (LOP)

```
LOOP-NN.

   MOVE "LOOP-NN" TO STMT-LOC.
```

36. This section will generate the following for a NLP command.

```
GO TO EXIT-NN
```

where NN is the value of the current index + 1.

37. This section will generate code to write the RESULTS record to a file. (PIO)

    generate

    ```
    WRITE RESULTS-REC
    ADD 1 TO REC-COUNT
    ```

38. This section will generate the statement FIND CURRENT RECORD, to reset to the current of record type. (RCR)

    38.1 Generate

    FIND CURRENT recordid.

    where recordid is the value in RCS-RTID of structure AP-INFO-TABLE

    38.2 Generate the code to test the return status.

39. This section generates the following:

    GO TO EXIT-NN

    where NN is the value of the current index + 1.

40. This section will generate the commands to delete a record from a set. (RDS)

    40.1 Generate the following code:

    FIND NEXT Recname WITHIN Setname

    IF STATUS-OK

    ERASE Recname

    where Recname is the value in RD2-RTID and Setname the value in RD2-SETID of structure AP-INFO-TABLE

41. This section of the code will generate the statements to delete a record. (RDK)

41.1 Generate the MOVE to set up the key prior to the IDMS call as follows:

MOVE ISQL-VAR-NN TO destination.

where destination is determined as follows:

If the value in RK-RTID is spaces, generate:

RK-DFID

else generate

RK-DFID OF RK-RTID

where RK-RTID and RK-RTID are in structure AP-INFO-TABLE

41.2 Now generate the statements to delete the the record as follows:

FIND ANY Recid

IF STATUS-OK
ERASE Recid

where Recid is the value in RK-RTID

42. This section will generate the code to add a record to the database using direct access by key value.

42.1 Generate the MOVE statement to set up the key prior to the call as follows: (RIK)

MOVE IS-VAR-NN TO destination.

where destination is determined as follows:

If the value in RK-RTID is spaces, then generate:

RK-DFID for destination

else generate:

RK-DFID OF RK-RTID

where RK-RTID and RK-DFID are the record id and field id of the structure AP-INFO-TABLE

42.2 Now generate the insert

STORE RK-RTID

42.3 Generate code to test the return status.

43. This section will generate the code to do an insert neat command. (RIS)

43.1 Generate

STORE recname.

where recname is the value found in RI2-RTID of structure AP-INFO-TABLE

43.2 Generate the code to test the status code.

44. This section will generate the code to modify a record using direct access by key value. (RUK)

44.1 Generate the MOVE statement to set up the key to the call as follows:

MOVE ISQL-VAR-NN TO dbitem.

where dbitem is determined as follows:

If the value in RK-RTID is spaces, then generate:

RK-DFID FOR dbitem

else generate:

RK-DFID OF RK-RTID

where RK-DFID and RK-RTID are the data item and record names found in structure AP-INFO-TABLE

44.2 Generate the statement

MODIFY recname.

where recname is the value in RK-RTID

44.3 Generate the code to test the IDMS return status.

45. This section will generate the code to do a modify next statement. (RUS)

45.1 Generate the following:

MODIFY Recname

where Recname is the record name found in RU2-RTID of structure AP-INFO-TABLE

45.2 Generate the code to test the return status.

46. This section will generate the code required to remove a record from its participation as a member in a given set. (SD)

46.1 Generate the following:

DISCONNECT Recname FROM Setname.

where Recname is the value in SS-RTID and Setname is the value in GC-COMMAND of structure AP-INFO-TABLE

46.2 Generate the code to test the return status.

47. This section will generate the code required to insert a record as a member of a given set. (SI)

47.1 Generate the following:

CONNECT Recname INTO Setname.

where Recname and Setname are the values found SS-RTID and GC-COMMAND of structure AP-INFO-TABLE

47.2 Generate the code to test the return status.

48. CIF Command

Generate an IF statement to evaluate the qualifications in the NDML WHERE clause at the conceptual schema level.

```
Call "CDRPCIF" using
BOOLEAN-LIST,
CS-QUALIFY-LIST,
CS-ACTION LIST,
IS-ACTION-LIST,
PROFILE.
```

49. CAL Command

Generate code in the procedure division of the CODASYL request Processor to call a complex mapping algorithm for the transformation of data.

49.1 Generate code to bypass the call to the complex mapping algorithm if any null values.

Generate the following:

```
IF PARM-NULL-alg-modinst = 1
    GO TO alg-modinst.
```

where:

```
alg     = CAL-ALG-ID(GC-SPEC-PTR(GCINDEX))
modinst = CAL-MOD-INST(GC-SPEC-PTR(GC-INDEX))
```

49.2 Generate code to call the complex mapping algorithm.

```
CALL "alg"  USING PARM-alg-modinst-parmno
                          .
                          .
                          .
                          .
```

where:

```
alg     = CAL-ALG-ID(GC-SPEC-PTR-(GC-INDEX))
modinst = CAL-MOD-INST(GC-SPEC-PTR(GC-INDEX))
parmno  = 1 - n
n       = CAL-PARM-COUNT(GC-SPEC-PTR-(GC-INDEX))
```

49.3 Generate standard error handling code to process the return status from the complex mapping algorithm.

49.4 Generate the bypass point for the call to the complex mapping algorithm.

Generate the following:

alg-modinst.

where:

```
alg     = CAL-ALG-ID(GC-SPEC-PTR(GC-INDEX))
modinst = CAL-MOD-INST(GC-SPEC-PTR-(GC-INDEX))
```

50. EP Command

Generate the end of a loop construct of the CODASYL Request Processor.

Generate the following:

```
EXIT-nn.
```

where:

```
nn = GC-SPEC-PTR + 1
```

51. GF2 Command

Generate code to move a set value to a local variable declared in the working storage section.

Generate the following:

```
MOVE "set-value" TO ISQside-VAR-nn.
```

where:

```
set-value = FG2-VALUE(GC-SPEC-PTR(GC-INDEX))
side      = FG2-SIDE(GC-SPEC-PTR(GC-INDEX))
nn        = FG2-ISQ-PTR(GC-SPEC-PTR(GC-INDEX))
```

52. GIF Command

Generate code to determine if a retrieved data field contains a null value and establish the null flag for this result field. If the retrieved field does not contain a null value, generate code to move the retrieved data field which is in internal format to

the results file which is in conceptual format. Generate code to perform numeric checks on retrieved data to avoid an abnormal end of processing.

52.1 Generate the following procedure division code if the retrieved data field has a character data type (RF1-TYPE(GC-SPEC-PTR(GC-INDEX)) = "C"):

```
IF D-dfno = character-null-value
   MOVE 1 TO RES-NULL-nn
   MOVE ZEROES TO RES-nn
IF D-dfno not = character-null-value
   MOVE 0 TO RES-NULL-nn
   MOVE D-dfno TO RES-nn.
```

where:

dfno = RF1-DFNO(GC-SPEC-PTR(GC-INDEX))
character-null-value =
   CHARACTER-NULL-VALUE from input parameter
nn   = RF1-IS-PTR(GC-SPEC-PTR(GC-INDEX))

52.2 Generate the following procedure division code if the retrieved data field has a numeric data type
(RF1-TYPE(GC-SPEC-PTR(GC-INDEX)) NOT = "C"):

```
IF D-dfno NOT NUMERIC
   STRING "DATA CONTAINED IN DATA FIELD"
           RF1-DFID (GC-SPEC-PTR (GC-INDEX))
          "FOR RECORD"
           RF1-RTID (GC-SPEC-PTR (GC-INDEX))
          "IS NON NUMERIC"
   DELIMITED BY SIZE
   INTO MESG-DESC
   PERFORM PROCESS-ERROR
   MOVE 1 TO RES-NULL-nn
   MOVE ZEROES TO RES-nn
ELSE
   IF D-dfno = numeric-null-value
      MOVE 1 TO RES-NULL-nn
      MOVE ZEROES TO RES-nn
   ELSE
      MOVE 0 TO RES-NULL-nn
      MOVE D-dfno TO RES-nn.
```

where:

```
dfno = RF1-DFNO(GC-SPEC-PTR(GC-INDEX))
numeric-null-value =
       NUMERIC-NULL-VALUE from input parameter
nn   = RF1-IS-PTR(GC-SPEC-PTR(GC-INDEX))
```

53. IFC Command

Generate code for a condition of a data-field and local variable in an IF statement.

Generate the following procedure division statement:

```
condition ((D-dfno NOT = null-value AND
            ISQside-NULL NOT = 1)   AND
           (D-dfno op ISQside-VAR-nn))
```

where:

```
condition  = RS5-IF-OR (GC-SPEC-PTR(GC-INDEX))
dfno       = RS5-DFNO  (GC-SPEC-PTR(GC-INDEX))
side       = RS5-SIDE  (GC-SPEC-PTR(GC-INDEX))
op         = RS5-OP    (GC-SPEC-PTR(GC-INDEX))
             RS5-ISQ-PTR (GC-SPEC-PTR(GC-INDEX))
null-value = CHARACTER-NULL-VALUE from input
             parameter if:
             RS5-TYPE(GC-SPEC-PTR(GC-INDEX)) = "C"
           = NUMERIC-NULL-VALUE from input parameter
      if:
          RS5-TYPE(GC-SPEC-PTR(GC-INDEX)) NOT = "C"
```

54. IFX Command

Generate code to determine if the current index is greater than the maximum index value.
Generate the following procedure division statement:

```
IF INDEX-dfno > INDEX-dfno-MAX
```

where:

```
dfno = OC4-INDEX-DFNO(GC-SPEC-PTR(GC-INDEX))
```

55. IF1 Command

Generate code for the following conditional test of a looping construct for key values:

```
ADD 1 TO LOOP-COUNT.
IF LOOP-COUNT > max
    GO TO EXIT-nn
```

where:

```
max = RK1-LOOP-MAX(GC-SPEC-PTR(GC-INDEX))
nn  = GC-SPEC-PTR(GC-INDEX) - 1
```

56. IF2 Command

Generate code for the following conditional test of a looping construct for repeating data fields:

```
IF LOOP-COUNT = n
```

where:

```
n = RK2-LOOP-COUNT(GC-SPEC-PTR(GC-INDEX))
```

57. IIF Command

Generate an IF statement to evaluate the qualifications in the NDML WHERE clause at the internal schema level.

```
Call CDRPIIF using
SUBTRANS-BOOLEAN-LIST,
IS-QUALIFY-LIST,
Procedure file,
SUBTRANS-ID,
CHARACTER-NULL-VALUE,
NUMERIC-NULL-VALUE,
RETURN-STATUS.
```

58. IX1

Generate code to increment the Data Field Index by 1. Generate the following procedure division statement:

```
SET INDEX - dfno UP BY 1.
```

where:

```
dfno = OC4-INDEX-DFNO(GC-SPEC-PTR(GC-INDEX)).
```

59. MR1 Command

Generate code to move a record from a schema area to a working storage data definition of the record.

Generate the following procedure division statement:

```
MOVE rtid TO R-rtno.
```

where:

```
rtid = MR1-RTID(GC-SPEC-PTR(GC-INDEX))
rtno = MR1-RTNO(GC-SPEC-PTR(GC-INDEX))
```

60. MR2 Command

Generate code to move a working storage data definition of a record to a record in the schema area.

Generate the following procedure division statement:

```
MOVE R-rtno TO rtid.
```

where:

```
rtid = MR2-RTID(GC-SPEC-PTR(GC-INDEX))
rtno = MR2-RTNO(GC-SPEC-PTR(GC-INDEX))
```

61. MV Command

Generate code to move a conceptual schema update value to an internal schema data field. If this field has not been mapped from the entity class being updated, null values will be moved to the data field.

If FUS-NULL(GC-SPEC-PTR(GC-INDEX)) = 0 generate:

```
MOVE MSG-VARI-nn TO D-dfno
```

where:

```
nn   = FUS-IS-PTR(GC-SPEC-PTR(GC-INDEX))
dfno = FUS-DFNO(GC-SPEC-PTR(GC-INDEX)) NOT = 0
```

IF FUS-NULL(GC-SPEC-PTR(GC-INDEX)) NOT = 0 generate:

```
MOVE null-value TO D-dfno
```

where:

```
dfno       = FUS-DFNO(GC-SPEC-PTR(GC-INDEX))
null-value = CHARACTER-NULL-VALUE from input
   parameter if:
          FUS-TYPE (GC-SPEC-PTR(GC-INDEX)) = "C"
           = NUMERIC-NULL-VALUE from input
             parameter if:
            FUS-TYPE (GC-SPEC-PTR(GC-INDEX)) NOT
   = "C"
```

62. MVI Command

Generate code to move variable containing the number of occurrences or occurs depending on value to a local variable.

```
MOVE ISQL-VAR-nn TO INDEX-dfno-MAX
```

where:

```
nn   = OC2-ISQ-PTR (GC-SPEC-PTR(GC-INDEX))
dfno = OC2-INDEX-DFNO (GC-SPEC-PTR(GC-INDEX))
```

63. MVK Command

Generate code to move a key field value to a key field of a record.

```
MOVE ISQL-VAR-nn TO dfid OF rtid.
```

where:

```
nn   = RK2-RK-INDEX (GC-SPEC-PTR(GC-INDEX))
dfid = RK2-DFID (GC-SPEC-PTR(GC-INDEX))
rtid = RK2-RTID (GC-SPEC-PTR(GC-INDEX))
```

64. MVM Command

Generate code to move data field or value containing the number of occurrences or occurs depending on value to a local variable:

IF OC3-MAX-OCCURS (GC-SPEC-PTR(GC-INDEX)) > 0 generate:

```
MOVE nn TO INDEX-indexdfno-MAX
```

otherwise generate:

```
MOVE D-dfno TO INDEX-indexdfno-MAX.
```

where:

```
nn        = OC3-MAX-OCCURS (GC-SPEC-PTR(GC-INDEX))
indexdfno = OC3-INDEX-DFNO (GC-SPEC-PTR(GC-INDEX))
dfno      = OC3-OCCURS-DEP-DFNO
        (GC-SPEC-PTR(GC-INDEX))
```

65. MVS Command

Generate code to move a conceptual schema search value to a local variable in internal schema format.

```
MOVE MSG-VAR-nn TO ISQL-VAR-nn
```

where:

nn = MVS-ISQ-PTR (GC-SPEC-PTR(GC-INDEX))

66. MVX Command

Generate code to move an indexed field to the results record.

```
MOVE D-dfno (INDEX-dfno1,INDEX-dfno2,INDEX-dfno3) TO
        RES-nn
```

where:

```
dfno  = OC5-DFNO    (GC-SPEC-PTR(GC-INDEX))
nn    = OC5-IS-PTR (GC-SPEC-PTR(GC-INDEX))
dfno1, dfno2, dfno3 are used depending on the
     value of
OC5-NUM-INDEXES and are found in OC5-IDX-DFNO1,
OC5-IDX-DFNO2, OC5-IDX-DFNO3
```

66a. MVY Command

Generate code to move null values to a repeating data field for an insert of a record.

```
MOVE null-value TO
    D-dfno (INDEX-dfno1, INDEX-dfno2, INDEX-dfno3)
```

where:

```
    dfno = OC6-DFNO(GC-SPEC-PTR(GC-INDEX))
    dfno1, dfno2, dfno3 are used depending on the
value of     OC6-NUM-INDEXES and are found in
```

```
OC6-IDX-DFNO1,     OC6-IDX-DFNO2, OC6-IDX-DFNO3
    null-value = CHARACTER-NULL-VALUE from input
                 parameter if:
                 OC6-TYPE(GC-SPEC-PTR(GC-INDEX)) =
"C"
               = NUMERIC-NULL-VALUE from input
                 parameter if:
                 OC6-TYPE(GC-SPEC-PTR(GC-INDEX)) NOT
"C"
```

67. MV2 Command

Generate code to initialize the local variable controlling the number of iterations through the construct for multiple values of a key.

```
    MOVE ZERO TO LOOP-COUNT.
```

68. MV3 Command

Generate code to move a conceptual schema update value to
the input parameter of a complex mapping algorithm.
Generate code to initialize the null flag for the algorithm.

```
   MOVE MSG-VARI-nn TO
        PARM-alg-modinst-parmno.
   MOVE ZERO TO PARM-NULL-alg-modinst.
```

where:

```
   nn      = FG3-IS-PTR    (GC-SPEC-PTR(GC-INDEX))
   alg     = FG3-ALG-ID    (GC-SPEC-PTR(GC-INDEX))
   modinst = FG3-MOD-INST  (GC-SPEC-PTR(GC-INDEX))
   parmno  = FG3-PARM-NO   (GC-SPEC-PTR(GC-INDEX))
```

69. MV4 Command

Generate code to move a constant value to an input parameter of a complex mapping algorithm.

```
   MOVE constant-value TO PARM-alg-modinst-parmno
```

where:

```
constant-value = FG4-CONSTANT (GC-SPEC-PTR(GC-INDEX))
alg            = FG4-ALG-ID   (GC-SPEC-PTR(GC-INDEX))
modinst        = FG4-MOD-INST (GC-SPEC-PTR(GC-INDEX))
parmno         = FG4-PARM-NO  (GC-SPEC-PTR(GC-INDEX))
```

70. MV5 Command

Generate code to move an output parameter of a complex mapping algorithm to a working storage record description structure. Generate code to set the null flag indicator for this algorithm to zero.

```
MOVE PARM-alg-modinst-parmno TO rtid-AREA.
MOVE ZERO TO PARM-NULL-alg-modinst.
```

where:

```
alg      = FU2-ALG-ID   (GC-SPEC-PTR(GC-INDEX))
modinst  = FU2-MOD-INST (GC-SPEC-PTR(GC-INDEX))
parmno   = FU2-PARM-NO  (GC-SPEC-PTR(GC-INDEX))
rtid     = FU2-RTID     (GC-SPEC-PTR(GC-INDEX))
```

71. MV6 Command

Generate code to move an output parameter of a complex mapping algorithm to a data field. Generate code to set the null flag indicator for this algorithm to zero.

```
MOVE PARM-alg-modinst-parmno TO D-dfno.
MOVE ZERO TO PARM-NULL-alg-modinst.
```

where:

```
alg      = FU1-ALG-ID   (GC-SPEC-PTR(GC-INDEX))
modinst  = FU1-MOD-INST (GC-SPEC-PTR(GC-INDEX))
parmno   = FU1-PARM-NO  (GC-SPEC-PTR(GC-INDEX))
dfno     = FU1-DFNO     (GC-SPEC-PTR(GC-INDEX))
```

72. MV7 Command

Generate code to move a non-null data field to an input parameter of a complex mapping algorithm. Generate code to set the null flag indicator for this algorithm based on the data field value. Generate code to perform numeric checks on retrieved data fields to avoid an abnormal end of processing.

Generate the following if FU3-TYPE(GC-SPEC-PTR(GC-INDEX)) = "C":

```
IF D-dfno = null-value
   MOVE 1 TO PARM-NULL-alg-modinst
   MOVE ZEROES TO PARM-alg-modinst-parmno

IF D-dfno NOT = null-value
   MOVE 0 TO PARM-NULL-alg-modinst
   MOVE D-dfno TO PARM-alg-modinst-parmno
```

Generate the following if FU3-TYPE(GC-SPEC-PTR(GC-INDEX)) NOT = "C"

```
IF D-dfno NOT NUMERIC
    STRING "DATA CONTAINED IN DATA FIELD"
            FU3-DFID (GC-SPEC-PTR (GC-INDEX))
           "FOR RECORD"
            FU3-RTID (GC-SPEC-PTR(GC-INDEX))
           "IS NON NUMERIC"
    DELIMITED BY SIZE
    INTO MESG-DESC
    PERFORM PROCESS-ERROR
    MOVE 1 TO RES-NULL-nn
    MOVE ZEROES TO RES-nn
ELSE
    IF D-dfno = null-value
       MOVE 1 TO PARM-NULL-alg-modinst
       MOVE ZEROES TO PARM-alg-modinst-parmno
    ELSE
       MOVE 0 TO PARM-NULL-alg-modinst
       MOVE D-dfno TO PARM-alg-modinst-parmno
```

where:

```
nn         = FU3-IS-PTR   (GC-SPEC-PTR(GC-INDEX))
dfno       = FU3-DFNO     (GC-SPEC-PTR(GC-INDEX))
alg        = FU3-ALG-ID   (GC-SPEC-PTR(GC-INDEX))
modinst    = FU3-MOD-INST (GC-SPEC-PTR(GC-INDEX))
null-value = CHARACTER-NULL-VALUE from input parameter if:
               FU3-TYPE (GC-SPEC-PTR(GC-INDEX)) = "C"
           = NUMERIC-NULL-VALUE from input parameter if:
               FU3-TYPE (GC-SPEC-PTR(GC-INDEX)) NOT = "C"
```

73. MV8 Command

Generate code to move the contents of a record to an input parameter of a complex mapping algorithm. Generate code to set the null flag indicator for the algorithm to zero.

```
MOVE rtid-AREA TO PARM-alg-modinst-parmno.
MOVE 0 TO PARM-NULL-alg-modinst.
```

where:

```
rtid    = FU4-RTID      (GC-SPEC-PTR(GC-INDEX))
alg     = FU4-ALG-ID    (GC-SPEC-PTR(GC-INDEX))
modinst = FU4-MOD-INST  (GC-SPEC-PTR(GC-INDEX))
parmno  = FU4-PARM-NO   (GC-SPEC-PTR(GC-INDEX))
```

74. NXS Command

Generate a COBOL "NEXT SENTENCE" statement in the procedure division of the Request Processor.

```
NEXT SENTENCE
```

75. OU1 Command

This command requires no code to be generated.

76. OU2 Command

Generate code to move a set value to the results record.

```
MOVE set-value TO RES-nn
```

where:

```
set-value = RF2-VALUE (GC-SPEC-PTR(GC-INDEX))
nn        = RF2-PTR   (GC-SPEC-PTR(GC-INDEX))
```

77. OU3 Command

Generate code to move the output parameter of a complex algorithm to the results record and establish the null flag for this result field.

```
IF PARM-NULL-alg-modinst = 1
   MOVE 1 TO RES-NULL-nn
   MOVE ZEROS TO RES-nn

IF PARM-NULL-alg-modinst = 0
   MOVE 0 TO RES-NULL-nn
   MOVE PARM-alg-modinst-parmno TO RES-nn
```

where:

```
alg      = RF3-ALG-ID                (GC-SPEC-PTR(GC-INDEX))
modinst  = RF3-MOD-INST              (GC-SPEC-PTR(GC-INDEX))
nn       = RF3-IS-PTR                (GC-SPEC-PTR(GC-INDEX))
parmno   = RF3-PARM-NO(GC-SPEC-PTR (GC-INDEX))
```

78. OU4 Command

Generate code to move the output parameter of a complex mapping algorithm to a local variable and establish the null flag for this variable.

```
IF PARM-NULL-alg-modinst = 1
   MOVE 1 TO TAG-NULL-tagno
   MOVE ZEROES TO TAG-tagno
IF PARM-NULL-alg-modinst = 0
   MOVE 0 TO TAG-NULL-tagno
   MOVE PARM-alg-modinst-parmno TO TAG-tagno
```

where:

```
alg      = RF3-ALG-ID    (GC-SPEC-PTR(GC-INDEX))
modinst  = RF3-MOD-INST  (GC-SPEC-PTR(GC-INDEX))
tagno    = RF3-TAGNO     (GC-SPEC-PTR(GC-INDEX))
parmno   = RF3-PARM-NO   (GC-SPEC-PTR(GC-INDEX))
```

79. OU5 Command

Generate code to move a non-null data field to a local variable and establish the null flag for the variable. Generate code to perform numeric checks on retrieved data to avoid an abnormal end of processing.

Generate the following code if OU5-TYPE(GC-SPEC-PTR (GC-INDEX)) = "C":

```
IF D-dfno = null-value
   MOVE 1 TO TAG-NULL-tagno
   MOVE ZEROES TO TAG-tagno

IF D-dfno NOT = null-value
   MOVE 0 TO TAG-NULL-tagno
   MOVE D-dfno TO TAG-tagno
```

Generate the following code if OU5-TYPE(GC-SPEC-PTR(GC-INDEX)) NOT = "C":

```
IF D-dfno NOT NUMERIC
   STRING "DATA CONTAINED IN DATA FIELD"
            OU5 DFID (GC-SPEC-PTR (GC-INDEX))
```

```
                "FOR RECORD"
                OU5-RTID (GC-SPEC-PTR (GC-INDEX))
                "IS NON NUMERIC"
          DELIMITED BY SIZE
          INTO MESG-DESC
          PERFORM PROCESS-ERROR
          MOVE 1 TO TAG-NULL-tagno
          MOVE ZEROES TO TAG-tagno
       ELSE
          IF D-dfno = null-value
             MOVE 1 TO TAG-NULL-tagno
             MOVE ZEROES TO TAG-tagno
               ELSE
             MOVE 0 TO TAG-NULL-tagno
             MOVE D-dfno TO TAG-tagno
```

where:

```
dfno       = OU5-DFNO   (GC-SPEC-PTR(GC-INDEX))
tagno      = OU5-TAGNO  (GC-SPEC-PTR(GC-INDEX))
null-value = CHARACTER-NULL-VALUE from input parameter if:
               OU5-TYPE (GC-SPEC-PTR(GC-INDEX)) = "C"
               NUMERIC-NULL-VALUE from input parameter if:
                OU5-TYPE (GC-SPEC-PTR(GC-INDEX)) NOT = "C"
```

80. SXI Command

Generate code to establish the value of an index field from
a local variable.

```
   SET INDEX-dfno TO ISQL-VAR-nn
```

where:

```
   dfno = OC2-INDEX-DFNO (GC-SPEC-PTR(GC-INDEX))
   nn   = OC2-ISQ-PTR    (GC-SPEC-PTR(GC-INDEX))
```

81. SX1 Command

Generate code to initialize an index field to value of 1.

```
   SET INDEX-dfno TO 1.
```

where:

```
   dfno = OC1-INDEX-DFNO (GC-SPEC-PTR(GC-INDEX))
```

82. UIF Command

    Generate an IF statement to evaluate the union discriminators in the NDML transaction.

83. Generate code to process errors and send results/status to the RP main program. Call "CDMACR" using QPG06 as the macro name.

84. Combine the two sequential work files used in generating the Request Processor.

    Call "CDCWF" with the following:

        FILE-NAME1
        FILE-NAME2

85. Return to CDP13 to continue generating code to satisfy the NDML request.

24.3 OUTPUT

The CODASYL RP program will generate the following output:

1. The DATA DIVISION of each RP will contain the following:

   1.1 A FILE SECTION containing the file description of the output results file.

   1.2 WORKING STORAGE section containing the data definitions for the CODASYL function codes, search arguments, etc. Variables to be used by the RP to perform internal functions.

2. The PROCEDURE DIVISION of each RP will consist of the following:

   2.1 Code to set up the results file.

   2.2 Code to convert from CS/IS and IS/CS.

   2.3 Code to process the requested NDML function against the CODASYL database. This may require the generation of code to navigate through the database.

   2.4 Code to test the return status of each database call.

2.5 Standard error message generation.

3. Name of the file containing the generated request processor.

4. Status of the function processing.

GC	APITABLE	1. CODE
CIF	CIF	1. IF statement for WHERE clause qualification (conceptual schema)
CAL	CAL	1. IF PARM-NULL-alg-modinst = 1 GO TO alg-modinst 2. CALL "alg-id" USING PARM-alg-modinst-parmno . . . . 3. error handling 4. alg-modinst
EIF	N/A	1. .(period)
ELP	NN	1. GO TO LOOP-nn. 2. EXIT-nn.
ELS	N/A	1. ELSE
EP	-	1. EXIT-nn. (+1)
FFA	RAS	1. FIND FIRST REC-NAME WITHIN AREA-NAME. 2. PERFORM dbms-STATUS. 3. IF OK-STATUS GET REC-NAME PERFORM dbms-STATUS.
FFM	SST	1. FIND FIRST REC-NAME WITHIN SET-NAME. 2. PERFORM dbms-STATUS. 3. IF OK-STATUS GET-REC-NAME PERFORM dbms-STATUS

FFR	RK	1. FIND FIRST REC-NAME. 2. PERFORM dbms-STATUS. 3. IF OK-STATUS GET REC-NAME PERFORM dbms-STATUS.
FNA	RAS	1. FIND NEXT REC-NAME WITHIN AREA-NAME. 2. PERFORM dbms-STATUS. 3. IF OK-STATUS GET REC-NAME PERFORM dbms-STATUS.
FNM	SST	1. FIND NEXT REC-NAME WITHIN SET-NAME. 2. PERFORM dbms-STATUS. 3. IF OK-STATUS GET REC-NAME. PERFORM dbms-STATUS.
FNR	RK	1. FIND NEXT REC-NAME. 2. PERFORM dbms-STATUS. 3. IF OK-STATUS GET REC-NAME PERFORM dbms-STATUS.
FOW	SST	1. FIND OWNER WITHIN SET-NAME. 2. PERFORM dbms-STATUS. 3. IF OK-STATUS GET REC-NAME PERFORM dbms-STATUS.
GF2	FG2	1. MOVE "set-value" TO ISQL-VAR-nn. or MOVE "set-value" TO ISQR-VAR-nn.

GIF	RF1	1. IF D-dfno = null-value MOVE 1 TO RES-NULL-nn MOVE ZEROES TO RES-nn 2. IF D-dfno NOT = null-value MOVE 0 TO RES-NULL-nn MOVE D-dfno TO RES-nn
GIO	FG1	1. IF D-dfno = null-value MOVE 1 TO ISQL-NULL-nn MOVE ZEROES TO ISQL-VAR-nn 2. IF D-dfno NOT = null-value MOVE 0 TO ISQL-NULL-nn MOVE D-dfno TO ISQL-VAR-nn or 1. IF D-dfno = null value MOVE 1 TO ISQR-NULL-nn MOVE 0 TO ISQR-VAR-nn 2. IF D-dfno = null-value MOVE 0 TO ISQR-NULL-nn MOVE D-dfno TO ISQR-VAR-nn
GI1	-	NO-OP
GI2	-	NO-OP
IF	IT2	1. IF "set-value" OP ISQL-VAR-nn or IF "set-value" OP IS-VAR-nn
IFC	RS5	1. IF ((D-dfno NOT = null-value AND ISQL-NULL = 1) AND (D-dfno op ISQL-VAR-nn)) or OR ((D-dfno NOT = null-value AND ISQL-NULL NOT = 1) AND (D-dfno op ISQL-VAR-nn)) or IF ((D-dfno NOT = null-value AND ISQR-NULL NOT = 1) AND

		(D-dfno op ISQR-VAR-nn)) or OR ((D-dfno NOT = null-value AND ISQR-NULL NOT = 1) AND (D-dfno op ISQR-VAR-nn))
IFX	0C4	1. IF INDEX-dfno > INDEX-dfno-MAX
IF1	RK1	1. ADD 1 TO LOOP-COUNT 2. IF LOOP-COUNT > MAX GO TO EXIT-(n-1).
IF2	RK2	1. IF LOOP-COUNT = n
IIF	CIF	1. IF statement for WHERE clause qualification (internal schema)
IN1	RS1	1. IF NOT (D-dfno1 = null-value AND D-DFNO2 = null-value) AND NOT (D-dfno1 OP D-dfno2)
IN2	RS4	1. IF NOT (D-dfno = null-value AND ISQL-NULL-nn = 1) AND NOT (D-dfno op ISQL-VAR-nn) or IF NOT (D-dfno = null-value AND ISQR-NULL-nn = 1) AND NOT (D-dfno op ISQR-VAR-nn)
IN3	RS3	1. IF NOT ("set-value" op ISQL-VAR-nn) or IF NOT ("set-value" op ISQR-VAR-nn)

IRF	SST	1. IF ((OK) AND NOT (EOS OR EOA OR EOC OR EOO))
IRN	SST	1. IF ((EOS OR EOA OR EOC OR EOO) AND OK))
LOP	NN	1. LOOP-nn.
MR1	MR1	1. MOVE rtid TO R-rtno.
MR2	MR2	1. MOVE R-rtno TO rtid.
MV	FUS	1. MOVE MSG-VARI-nn TO D-dfno or MOVE null-value TO D-dfno
MVI	OC2	1. MOVE ISQL-VAR-nn TO INDEX-dfno-MAX.
MVK	RK2	1. MOVE ISQL-VAR-nn TO D-dfno
MVM	OC3	1. MOVE nn TO INDEX-dfno-MAX. or 2. MOVE D-dfno TO INDEX-dfno-MAX.
MVS	MVS	1. MOVE MSG-VAR-nn TO ISQL-VAR-nn.
MVX	OC5	1. MOVE D-dfno (INDEX-dfno,...) TO RES-nn.
MVY	MVY	1. MOVE null-value TO D-dfno (INDEX-dfno...)
MVZ	-	1. MOVE ZERO TO LOOP-COUNT.

MV3	FG3	1. MOVE MSG-VAR-nn TO PARM-alg-modinst-parmno. 2. MOVE ZERO TO PARM-NULL-alg-modinst.
MV4	FG4	1. MOVE constant-value TO PARM-alg-modinst-parmno.
MV5	FU2	1. MOVE PARM-alg-modinst-parmno TO rtid-AREA 2. MOVE ZERO TO PARM-NULL-alg-modinst.
MV6	FU1	1. MOVE PARM-alg-modinst-parmno TO D-dfno 2. MOVE ZERO TO PARM-NULL-alg-modinst.
MV7	FU3	1. IF D-dfno = null-value MOVE 1 TO PARM-NULL-alg-modinst MOVE ZEROES TO PARM-alg-modinst-parmno 2. IF D-dfno NOT = null-value MOVE 0 TO PARM-NULL-alg-modinst MOVE D-dfno TO PARM-alg-modinst-parmno
MV8	FU4	1. MOVE rtid-AREA TO PARM-alg-modinst-parmno 2. MOVE 0 TO PARM-NULL-alg-modinst.
NLP	NN	1. GO TO EXIT-nn. (+1)
NXS	-	1. NEXT SENTENCE

OU1	-	NO-OP
OU2	RF2	1. MOVE "set-value" TO RES-nn.
OU3	RF3	1. IF PARM-NULL-alg-modinst = 1 MOVE 1 TO RES-NULL-nn MOVE ZEROES TO RES-nn 2. IF PARM-NULL-alg-modinst = 0 MOVE 0 TO RES-NULL-nn MOVE PARM-alg-modinst-parmno TO RES-nn
OU4	RF3	1. IF PARM-NULL-alg-modinst = 1 MOVE 1 TO TAG-NULL-tagno MOVE ZEROES TO TAG-tagno 2. IF PARM-NULL-alg-modinst = 0 MOVE 0 TO TAG-NULL-tagno MOVE PARM-alg-modinst-parmno TO TAG-tagno
OU5	OU5	1. IF D-dfno = null-value MOVE 1 TO TAG-NULL-tagno MOVE ZEROES TO TAG-tagno 2. IF D-dfno NOT = null-value MOVE 0 TO TAG-NULL-tagno MOVE D-dfno TO TAG-tagno
PIO	N/A	1. WRITE RESULTS-REC. 2. ADD 1 TO REC-COUNT.
RCR		1. FIND CURRENT REC-NAME 2. PERFORM dbms-STATUS 3. IF OK-STATUS GET REC-NAME PERFORM dbms-STATUS.
RDK	RD1	1. FIND ANY REC-NAME. 2. PERFORM dbms-STATUS. 3. IF OK-STATUS ERASE REC-NAME PERFORM dbms-STATUS 4. ELSE STRING "TRIED TO DELETE

```
                                NON-EXISTENT RECORD"
                                   DELIMITED BY SIZE
                                   INTO MESG-DESC

RDS         RD2                 1.  FIND NEXT REC-NAME
                                      WITHIN SET-NAME.
                                2.  PERFORM dbms-STATUS
                                3.  IF OK-STATUS
                                      ERASE REC-NAME
                                      PERFORM VAX-11-STATUS
                                    4.  ELSE
                                      STRING "TRIED TO DELETE
                                         NON-EXISTING RECORD"
                                       DELIMITED BY SIZE
                                      INTO MESG-DESG.

RIK         RI1                 1.  STORE REC-NAME.
                                      PERFORM VAX-11-STATUS.

RIS         RI2                 1.  STORE REC-NAME.
                                                PERFORM VAX-11-STATUS.

RUK         RU1                 1.  MODIFY REC-NAME.
                                                PERFORM VAX-11-STATUS.

RUS         RU2                 1.  MODIFY REC-NAME.
                                                PERFORM VAX-11-STATUS.

SD          SD                  1.  DISCONNECT REC-NAME FROM SET-NAME
                                                PERFORM VAX-11-STATUS.

SI          SI                  1.  CONNECT REC-NAME INTO SET-NAME.
                                                PERFORM VAX-11-STATUS.

SXI         OC2                 1.  SET INDEX-dfno TO ISQL-VAR-nn

SX1         OC1                 1.  SET INDEX-dfno TO 1.
```

UIF	UIF	1. IF statement for WHERE clause qualification for union discriminators.
XLP	NN	1. GO TO EXIT-nn.

TABLE 24-2

VAX-11 REQUEST PROCESSOR MACROS

LIBRARY: VAX-11
MACRO: FFA

```
FIND FIRST REC-NAME
     WITHIN AREA-NAME.
PERFORM VAX-11-STATUS.
IF OK-STATUS
     GET REC-NAME
     PERFORM VAX-11-STATUS
```

TABLE 24-2

VAX-11 REQUEST PROCESSOR MACROS

LIBRARY: VAX-11
MACRO: FFM

```
FIND FIRST REC-NAME
     WITHIN SET-NAME.
PERFORM VAX-11-STATUS.
IF OK-STATUS
     GET REC-NAME
     PERFORM VAX-11-STATUS.
```

TABLE 24-2

VAX-11 REQUEST PROCESSOR MACROS

LIBRARY: VAX-11
MARCO: FFR

```
FIND FIRST REC-NAME
     USING ITEM-NAME.
PERFORM VAX-11-STATUS.
IF OK-STATUS
     GET REC-NAME
     PERFORM VAX-11-STATUS.
```

TABLE 24-2

VAX-11 REQUEST PROCESSOR MACROS

LIBRARY: VAX-11
MACRO: FNA

```
FIND NEXT REC-NAME
     WITHIN AREA-NAME.
PERFORM VAX-11-STATUS.
IF OK-STATUS
     GET REC-NAME
     PERFORM VAX-11-STATUS
```

TABLE 24-2

VAX-11 REQUST PROCESSOR MACROS

LIBRARY: VAX-11
MACRO: FNM

```
FIND NEXT REC-NAME
    WITHIN SET-NAME.
PERFORM VAX-11-STATUS.
IF OK-STATUS
    GET REC-NAME
    PERFORM VAX-11-STATUS.
```

TABLE 24-2

VAX-11 REQUEST PROCESSOR MACROS

LIBRARY: VAX-11
MACRO: FNR

```
FIND NEXT REC-NAME
    USING ITEM-NAME.
PERFORM VAX-11-STATUS.
IF OK-STATUS
    GET REC-NAME
    PERFORM VAX-11-STATUS.
```

TABLE 24-2

VAX-11 REQUEST PROCESSOR MACROS

LIBRARY: VAX-11
MACRO: FOW

```
FIND OWNER
    WITHIN SET-NAME.
PERFORM VAX-11-STATUS.
IF OK-STATUS
    GET REC-NAME
    PERFORM VAX-11-STATUS.
```

TABLE 24-2

VAX-11 REQUST PROCESSOR MACROS

LIBRARY: VAX-11
MACRO: IRF

IF ((OK) AND NOT (EOS OR EOA OR EOC OR EOO))

TABLE 24-2

VAX-11 REQUEST PROCESSOR MACROS

LIBRARY: VAX-11
MACRO: IRN

IF ((EOS OR EOA OR EOC OR EOO) AND (OK))

TABLE 24-2

VAX-11 REQUEST PROCESSOR MACROS

LIBRARY: VAX-11
MACRO: QPG01

```
IDENTIFICATION DIVISION
PROGRAM-ID.  QQQQQ.
ENVIRONMENT DIVISION.
```

TABLE 24-2

VAX-11 REQUEST PROCESSOR MACROS

LIBRARY: VAX-11
MACRO: QPG02

```
    01 RET-STATUS    PIC X(5) VALUE SPACES.
    01 MODULE-NAME   PIC X(10) VALUE IS "QQQQQ".
    01 STMT-LOC      PIC X(30).
    01 MESG-DESC     PIC X(60).
    01 MY-HOST       PIC XXX   VALUE SPACE.
    01 COPY ERRCDM OF IISSLIB.
       COPY ERRFS OF IISSCLIB.
       01  FCB1                PIC S9(9) COMP.
       01  RECORD-LENGTH       PIC S9(9) COMP.
       01  ACCESS-CODE         PIC X.
       01  NUMBER-OF-RECORDS   PIC S9(9) COMP VALUE 2000.
       01  ERROR-FLAG          PIC X(5).
    01 DBMS-STATUS  PIC S9(9).
       88  EOA VALUE 2654548.
       88  EOS VALUE 2654548.
       88  EOC VALUE 2654548.
       88  EOO VALUE 2654548.
       88  OK-STATUS VALUE 1.
       88  OK VALUE 1 2654548.
       88  NON-FATAL VALUE 1 2654548.
    01 COBFILE.
       03 RESFILE  PIC X(30).
    01 FILE-OPEN    PIC 9.
       88  FILE-IS-OPEN    VALUE 1.
```

TABLE 24-2

VAX-11 REQUEST PROCESSOR MACROS

LIBRARY: VAX-11
MACRO: QPG03

```
01 MESSAGE-BODY-OUT.
   03  OUTFILE-NAME      PIC X(30).
   03  REC-COUNT         PIC 9(6).
   03  QP-STATUS         PIC X(5).
```

TABLE 24-2

VAX-11 REQUEST PROCESSOR MACROS

LIBRARY: VAX-11
MACRO: QPG05

```
    PROCEDURE DIVISION USING MESSAGE-BODY-IN
                             MESSAGE-BODY-OUT.
    DECLARATIVES.
    DB-DATABASE-EXCEPTIONS SECTION.  USE FOR DB-EXCEPTION.
    DB-ERROR-ROUTINE.
    END-DECLARATIVES.
    START-PROGRAM SECTION.
    START-RIGHT-HERE.
         MOVE ZERO                TO  FILE-OPEN.
         MOVE KES-SUCCESSFUL      TO  QP-STATUS.
         MOVE ZERO                TO  REC-COUNT.
         CALL "NAMFIL"  USING OUTFILE-NAME.
         IF OUTFILE-NAME = LOW-VALUE
            MOVE KES-NOFILE-NAME TO QP-STATUS
            MOVE OUTFILE-NAME  TO RESFILE.
```

TABLE 24-2

VAX-11 REQUEST PROCESSOR MACROS

LIBRARY: VAX-11
MACRO: QPG06

```
        GO TO 999-EOJ.
    PGM-ABORT.
        MOVE "ABORT" TO QP-STATUS.
    999-EOJ.
        MOVE RESFILE TO OUTFILE-NAME.
        EXIT PROGRAM.
    VAX-11-STATUS.
        MOVE DB-CONDITION TO VAX-11-STATUS.
        IF NOT OK
        STRING DBMS-STATUS DELIMITED BY SIZE
            "VAX-11 ERROR " DELIMITED BY SIZE
             STMT-LOC DELIMITED BY SIZE
        INTO MESG-DESC
             PERFORM PROCESS-ERROR
        GO TO PGM-ABORT.
    COPY ERRPRO OF IISSCLIB.
```

TABLE 24-2

VAX-11 REQUEST PROCESSOR MACROS

LIBRARY: VAX-11
MACRO: QPG21

```
FILE SECTION.
FD RESULTS
LABEL RECORDS ARE STANDARD
    VALUE OF ID IS RESFILE.
```

TABLE 24-2

VAX-11 REQUEST PROCESSOR MACROS

LIBRARY: VAX-11
MACRO: RCR

```
FIND CURRENT REC-NAME
PERFORM VAX-11-STATUS
IF OK-STATUS
     GET REC-NAME
     PERFORM VAX-11-STATUS.
```

TABLE 24-2

VAX-11 REQUEST PROCESSOR MACROS

LIBRARY: VAX-11
MACRO: RDK

```
FIND FIRST REC-NAME
     USING ITEM-NAME.
PERFORM VAX-11-STATUS.
IF OK-STATUS
     ERASE REC-NAME
     PERFORM VAX-11-STATUS
ELSE
     STRING "TRIED TO DELETE NON-EXISTENT RECORD"
          DELIMITED BY SIZE
          INTO MESG-DESC.
```

TABLE 24-2

VAX-11 REQUEST PROCESSOR MACROS

LIBRARY: VAX-11
MACRO: RDS

```
    PERFORM VAX-11-STATUS.
    IF OK-STATUS
         ERASE REC-NAME
         PERFORM VAX-11-STATUS
    ELSE
         STRING "TRIED TO DELETE NON-EXISTING RECORD"
                DELIMITED BY SIZE
                INTO MESG-DESC
```

TABLE 24-2

VAX-11 REQUEST PROCESSOR MACROS

LIBRARY: VAX-11
MACRO: RIK

```
STORE REC-NAME.
     PERFORM VAX-11-STATUS.
```

TABLE 24-2

VAX-11 REQUEST PROCESSOR MACROS

LIBRARY: VAX-11
MACRO: RIS

```
STORE REC-NAME.
PERFORM VAX-11-STATUS.
```

TABLE 24-2

VAX-11 REQUEST PROCESSOR MACROS

LIBRARY: VAX-11
MACRO: RUK

```
MODIFY REC-NAME.
     PERFORM VAX-11-STATUS.
```

TABLE 24-2

VAX-11 REQUEST PROCESSOR MACROS

LIBRARY: VAX-11
MACRO: RUS

```
MODIFY REC-NAME.
     PERFORM VAX-11-STATUS.
```

TABLE 24-2

VAX-11 REQUEST PROCESSOR MACROS

LIBRARY: VAX-11
MACRO: SD

DISCONNECT REC-NAME FROM SET-NAME
PERFORM VAX-11-STATUS

TABLE 24-2

VAX-11 REQUEST PROCESSOR MACROS

LIBRARY: VAX-11
MACRO: SI

```
CONNECT REC-NAME TO SET-NAME
     PERFORM VAX-11-STATUS
```

TABLE 24-2

IDMS REQUEST PROCESSOR MACROS

LIBRARY: IDMS
MACRO: FFA

```
FIND FIRST REC-NAME
     WITHIN AREA-NAME.
PERFORM IDMS-STATUS.
IF OK-STATUS
     GET REC-NAME
     PERFORM IDMS-STATUS
```

TABLE 24-2

IDMS REQUEST PROCESSOR MACROS

LIBRARY: IDMS
MACRO: FFM

```
FIND FIRST REC-NAME
     WITHIN SET-NAME.
PERFORM IDMS-STATUS.
IF OK-STATUS
     GET REC-NAME
     PERFORM IDMS-STATUS.
```

TABLE 24-2

IDMS REQUEST PROCESSOR MACROS

LIBRARY: IDMS
MARCO: FFR

```
FIND FIRST REC-NAME
     USING ITEM-NAME.
PERFORM IDMS-STATUS.
IF OK-STATUS
     GET REC-NAME
     PERFORM IDMS-STATUS.
```

TABLE 24-2

IDMS REQUEST PROCESSOR MACROS

LIBRARY: IDMS
MACRO: FNA

```
FIND NEXT REC-NAME
     WITHIN AREA-NAME.
PERFORM IDMS-STATUS.
IF OK-STATUS
     GET REC-NAME
     PERFORM IDMS-STATUS
```

TABLE 24-2

IDMS REQUEST PROCESSOR MACROS

LIBRARY: IDMS
MACRO: FNM

```
FIND NEXT REC-NAME
     WITHIN SET-NAME.
PERFORM IDMS-STATUS.
IF OK-STATUS
     GET REC-NAME
     PERFORM IDMS-STATUS.
```

TABLE 24-2

IDMS REQUEST PROCESSOR MACROS

LIBRARY: IDMS
MACRO: FNR

```
FIND NEXT REC-NAME
     USING ITEM-NAME.
PERFORM IDMS-STATUS.
IF OK-STATUS
     GET REC-NAME
     PERFORM IDMS-STATUS.
```

TABLE 24-2

IDMS REQUEST PROCESSOR MACROS

LIBRARY: IDMS
MACRO: FOW

```
FIND OWNER
     WITHIN SET-NAME.
PERFORM IDMS-STATUS.
IF OK-STATUS
     GET REC-NAME
     PERFORM IDMS-STATUS.
```

TABLE 24-2

IDMS REQUEST PROCESSOR MACROS

LIBRARY: IDMS
MACRO: IRF

IF ((OK) AND NOT (EOS OR EOA OR EOC OR EOO))

TABLE 24-2

IDMS REQUEST PROCESSOR MACROS

LIBRARY: IDMS
MACRO: IRN

IF ((EOS OR EOA OR EOC OR EOO) AND (OK))

TABLE 24-2

IDMS REQUEST PROCESSOR MACROS

LIBRARY: IDMS
MACRO: QPG01

```
IDENTIFICATION DIVISION.
PROGRAM-ID.  QQQQQ.
ENVIRONMENT DIVISION.
```

TABLE 24-2

IDMS REQUEST PROCESSOR MACROS

LIBRARY: IDMS
MACRO: QPG02

```
01   RET-STATUS     PIC X(5) VALUE SPACES.
01   MODULE-NAME    PIC X(10) VALUE IS "QQQQQ".
01   STMT-LOC       PIC X(30).
01   MESG-DESC      PIC X(60).
01   FCB1           PIC S9(9) COMP.
01   RECORD-LENGTH  PIC S9(9) COMP.
01   ACCESS-CODE    PIC X.
01   NUMBER-OF-RECORDS  PIC S9(9) COMP.
01   ERROR-FLAG     PIC X(5).
01   MY-HOST        PIC XXX   VALUE SPACE.
COPY ERRCDM OF IISSLIB.
COPY ERRFS OF IISSCLIB.
01   DBMS-STATUS  PIC X(4).
     88  EOA VALUE "0307".
     88  EOS VALUE "0307".
     88  EOC VALUE "0364" "0326" "0332" .
     88  EOO VALUE "0307".
     88  OK-STATUS VALUE "0000".
     88  OK VALUE "0307" "0364" "0326" "0332" "0000".
     88  NON-FATAL VALUE "0307" "0364" "0326" "0332" "0000".
01   COBFILE.
  03 RESFILE  PIC X(30).
01   FILE-OPEN     PIC 9.
     88  FILE-IS-OPEN    VALUE 1.
COPY QFAALC OF IISSCLIB.
01   OLDFILE       PIC X(8) VALUE "OLDFILE".
```

TABLE 24-2

IDMS REQUEST PROCESSOR MACROS

LIBRARY: IDMS
MACRO: QPG03

```
01  MESSAGE-BODY-OUT.
    03  OUTFILE-NAME      PIC X(30).
    03  REC-COUNT         PIC 9(6).
    03  QP-STATUS         PIC X(5).
```

TABLE 24-2

IDMS REQUEST PROCESSOR MACROS

LIBRARY: IDMS
MACRO: QPG05

```
    PROCEDURE DIVISION USING MESSAGE-BODY-IN
                             MESSAGE-BODY-OUT.
    START-RIGHT-HERE.
         MOVE ZERO              TO  FILE-OPEN.
         MOVE KES-SUCCESSFUL    TO  RET-STATUS.
         MOVE ZERO              TO  REC-COUNT.
         CALL "NAMFIL"  USING OUTFILE-NAME.
         IF OUTFILE-NAME = LOW-VALUE
            MOVE "UNABLE TO GENERATE FILE NAME" TO MESG-DESC
            MOVE KES-NOFILENAME TO RET-STATUS
            GO TO 999-EDJ.
         MOVE OUTFILE NAME      TO RESFILE.
         MOVE "W" TO ACCESS-CODE.
         CALL "OPNFIL" USING
                       FCB1,
                       ERROR-FLAG,
                       OUTFILE-NAME,
                       ACCESS-CODE,
                       RECORD-LENGTH,
                       NUMBER-OF-RECORDS.
         IF ERROR-FLAG NOT = KES-FILE-OK
            STRING "RESULTS FILE OPEN ERROR: " ERROR-FLAG
                   DELIMITED BY SIZE INTO MESG-DESC
            MOVE KES-OPEN-NOT-SUCCESSFUL TO RET-STATUS
            GO TO 999-EOJ.
```

TABLE 24-2

IDMS REQUEST PROCESSOR MACROS

LIBRARY: IDMS
MACRO: QPG06

```
           GO TO 999-EOJ.
      PGM-ABORT.
           MOVE "ABORT" TO QP-STATUS.
      999-EOJ.
           MOVE RESFILE TO OUTFILE-NAME.
           EXIT PROGRAM.
      IDMS-STATUS.
           MOVE ERROR-STATUS DBMS-STATUS.
           IF NOT OK
               STRING DBMS-STATUS DELIMITED BY SIZE
                  "IDMS ERROR " DELIMITED BY SIZE
                 STMT-LOC DELIMITED BY SIZE
              INTO MESG-DESC
               PERFORM PROCESS-ERROR
               GO TO PGM-ABORT.
           COPY ERRPRO OF IISSCLIB.
```

TABLE 24-2

IDMS REQUEST PROCESSOR MACROS

LIBRARY: IDMS
MACRO: QPG21

```
FILE SECTION.
FD RESULTS
LABEL RECORDS ARE STANDARD.
```

TABLE 24-2

IDMS REQUEST PROCESSOR MACROS

LIBRARY: IDMS
MACRO: RCR

```
FIND CURRENT REC-NAME
PERFORM IDMS-STATUS
IF OK-STATUS
     GET REC-NAME
     PERFORM IDMS-STATUS.
```

TABLE 24-2

IDMS REQUEST PROCESSOR MACROS

LIBRARY: IDMS
MACRO: RDK

```
     FIND FIRST REC-NAME
          USING ITEM-NAME.
     PERFORM IDMS-STATUS.
     IF OK-STATUS
          ERASE REC-NAME
          PERFORM IDMS-STATUS
     ELSE
          STRING "TRIED TO DELETE NON-EXISTENT RECORD"
               DELIMITED BY SIZE
               INTO MESG-DESC.
C
```

TABLE 24-2

IDMS REQUEST PROCESSOR MACROS

LIBRARY: IDMS
MACRO: RDS

```
     PERFORM IDMS-STATUS.
     IF OK-STATUS
          ERASE REC-NAME
          PERFORM IDMS-STATUS
     ELSE
          STRING "TRIED TO DELETE NON-EXISTING RECORD"
                 DELIMITED BY SIZE
                  INTO MESG-DESC
```

TABLE 24-2

IDMS REQUEST PROCESSOR MACROS

LIBRARY: IDMS
MACRO: RIK

```
STORE REC-NAME.
     PERFORM IDMS-STATUS.
```

TABLE 24-2

IDMS REQUEST PROCESSOR MACROS

LIBRARY: IDMS
MACRO: RIS

```
STORE REC-NAME.
PERFORM IDMS-STATUS.
```

TABLE 24-2

IDMS REQUEST PROCESSOR MACROS

LIBRARY: IDMS
MACRO: RUK

```
MODIFY REC-NAME.
     PERFORM IDMS-STATUS.
```

TABLE 24-2

IDMS REQUEST PROCESSOR MACROS

LIBRARY: IDMS
MACRO: RUS

```
MODIFY REC-NAME.
     PERFORM IDMS-STATUS.
```

TABLE 24-2

IDMS REQUEST PROCESSOR MACROS

LIBRARY: IDMS
MACRO: SD

```
DISCONNECT REC-NAME FROM SET-NAME
     PERFORM IDMS-STATUS
```

TABLE 24-2

IDMS REQUEST PROCESSOR MACROS

LIBRARY: IDMS
MACRO: SI

```
CONNECT REC-NAME TO SET-NAME
     PERFORM IDMS-STATUS
```

SECTION 25

FUNCTION PRE9.4 GENERATE TOTAL QUERY PROCESSOR

This function is a compile-time module (CDQPT) whose purpose is to generate the COBOL source code necessary to satisfy a NDML subtransaction request against a TOTAL database. The TOTAL RP extracts data through linkpaths using TOTAL as the DBMS and IBM as the host computer, and then reformats the data into a sequential file.

The major functions to be accomplished by the TOTAL Request Processor Generator are shown in Figure 15-1.

This document contains two reference tables. Table 15-1 is a cross reference of the code for the TOTAL calls that are generated from generic DML commands. Table 15-2 contains a listing of all code macros used by the TOTAL Request Processor Generator.

25.1 Inputs:

Inputs are the outputs from the following functions:

PRE4 - Transform ES/CS
PRE5 - Decompose CS NDML
PRE6 - Select IS Access Path
PRE7 - Transform IS Access Path/Generic

1. IS NDML Subtractions, represented by the IS-ACTION-LIST, IS-QUALIFY-LIST and COMPLEX-MAPPING-ALG-TABLE. See PRE5 for descriptions of these structures.

2. Conceptual schema formats.

   The Result Field Table contains the CS descriptions of the data fields requested by the NDML transaction and is defined in PRE5.

   The CS-QUALIFY-LIST contains the CS description of the variables and constants which are passed to the TOTAL RP at run-time and is described in PRE4.

   The CS-ACTION-LIST contains the CS descriptions of the data fields used by the NDML transaction and is described in PRE4.

3. Owner/Member Relationships

   The Set Table contains information about the sets that must be traversed in order to process the transaction and is described in PRE5.

4. Generic DML Commands

The GC-TABLE contains the generic DML commands to process the NDML request. The generic DML commands can be found in PRE7.

The Record Key Table contains the record keys for the TOTAL access path being processed.
The Access Path Selector Tables contain transaction data for the TOTAL access path being processed. Descriptions of these tables can be found in PRE6.

5. Subtransaction Variables

Database Descriptor Module (DBMOD) name.

```
01  QPGT-DBMOD-NAME        PIC X(6).
```

Database Identification Code

```
01  QPGT-DBID      PIC 9(5).
```

Subtransaction Identification Number

```
01  QPGT-SUBTRANS-ID       PIC 9(3).
```

Request Processor Program Name

```
01  QPGT-QP-NAME   PIC X(10).
```

Error File Name for User's AP

```
01  ERROR-FILE     PIC X(30).
```

Macro Library Name

```
01  LIBRARY-NAME   PIC X(30).
```

Current Host of Precompiler

```
01  CURRENT-HOST   PIC X(3).
```

Target Host of Request Processor

```
01  TARGET-HOST    PIC X(3).
```

Character Null Value for the Database

```
01  CHARACTER-NULL-VALUE   PIC X(30).
```

Numeric Null Value for the Database

```
01  NUMERIC-NULL-VALUE     PIC X(30).
```

6. CDM Requirements

The CDM entity class that must be accessed to generate a TOTAL Request Processor is:

DF USED AS SET LINKAGE - Data field set linkage mapping

25.2 Processing

The TOTAL Request Processor Generator (CDQPT) utilizes two sequential work files, to allow working-storage and procedure division statements to be constructed simultaneously. At the end of processing, the two files are combined.

Code which is common to all TOTAL Request Processors is generated utilizing the macro replacement utility CDMACR. Table 25-2 contains a listing of all code macros used by the TOTAL Request Processor Generator. Code specific to a TOTAL Request Processor is derived from information in the various tables, input parameters and CDM meta data.

Errors are reported using one of three error routines, depending on the type of error. User errors are reported by `RPTERR', system errors by `ERRPRO'. User errors consist of CDM meta data errors. All other types of errors, such as table overflows, DMBS errors, etc., are considered system errors. The generator may encounter both user and system errors. A Request Processor can encounter only system errors.

1. Initialize the Request Processor Generator internal tables and variables.

   Call routine GENFIL to establish the names of the two work files to be used during the generation of the Request Processor (QPGT-FILE-NAME1, QPGT-FILE-NAME2).

2. Create the RECORD-TABLE.

   2.1 Determine all unique record types in the IS-ACTION-LIST for this subtransaction.

   Add an entry to the RECORD-TABLE for each unique entry in the IS-ACTION-LIST with a matching subtransaction id.

```
Set RT-DBID = QPGT-DBID
```

   For each entry in the IS-ACTION-LIST with IS-SUBTRANS-ID equal to the subtransaction id of the RP (QPGT-SUBTRANS-ID), add an entry to the RECORD-TABLE if the record identification to be added does not already exist in the RECORD-TABLE:

```
RT-RTID = IS-RTID
RT-RTNO = IS-RTNO
RT-USED = RT-US
```

   2.2 Determine all unique record types in the left half of the IS-QUALIFY list for this subtransaction.

For each entry in the IS-QUALIFY-LIST with ISQ-SUBTRANS-IDL equal to the subtransaction id for the RP (QPGT-SUBTRANS-ID), add an entry to the RECORD-TABLE if the record identification to be added does not already exist in the RECORD-TABLE:

2.2.1 IF ISQ-RTIDL does not equal space

```
RT-RTNO = ISQ-RTNOL
RT-RTID = ISQ-RTIDL
RT-USED = RT-USED+1
```

2.2.2 If ISQ-RTIDL equals space, ignore the entry.

2.3 Determine all unique record types in the right half of the IS-QUALIFY list for this subtransaction.

For each entry in the IS-QUALIFY-LIST with ISQ-SUBTRANS-IDR equal to the subtransaction id of the RP (QPGT-SUBTRANS-ID), add an entry to the RECORD-TABLE if the record identification to be added does not already exist in the RECORD-TABLE:

2.3.1 If ISQ-RTIDR does not equal space

```
RT-RTNO = ISQ-RTNOR
RT-RTID = ISQ-RTIDR
RT-USED = RT-USED+1
```

2.3.2 IF ISQ-RTIDR equals space, ignore the entry.

2.4 Determine all unique record types in the SET-TABLE for this subtransaction.

For each entry in the SET-TABLE with ST-SUBTRANS-ID equal to the subtransaction id of the RP (QPGT-SUBTRANS-ID), add entries to the RECORD-TABLE for the record of the owner and member of the set. If these record ids already exist in the RECORD-TABLE, do not add entries.

```
RT-RTNO = ST-OWNER-ID
RT-RTID = ST-OWNER
RT-RTID = ST-MEMBER
RT-RTNO = ST-MEMBER-ID
```

Increment RT-USED for each entry added to the table.

2.5 Determine data fields used as a set linkage that will be required to process the NDML subtransaction.

For each entry in the GC-TABLE where GC-COMMAND = `FOW' (Find Owner of Set):

2.5.1 Access the SET-SPEC-TABLE using the GC-SPEC-PTR from the GC-TABLE entry and extract SS-SETID and RTID.

2.5.2 Search the SET TABLE for an entry matching current subtransaction id (QPGT-SUBTRANS-ID), database id (QPGT-DBID), SS-SETID and SS-RTID, ST-SETID in 2.5.1 with the ST-SUBTRANS-ID, ST-DBID, ST-SETID and ST-OWNER of the SET-TABLE entry. When a match is found, extract the record id of the member occurrence (ST-MEMBER).

2.5.3 Access the CDM entity class DF-USED AS SET LINKAGE with the current database id (QPGT-DBID), set id saved in 2.5.1 and the record id of the member occurrence saved in 2.5.2. Retrieve the data field used as the set linkage (DF-ID) and the linkage type (LINKAGE-TYPE). If the entity is not found, generate a NO-DF-SET-LINKAGE user error message and terminate processing. Verify that the linkage type variable has the value of "S". If this is not the case, generate an INVALID-SET-LINKAGE user error message and terminate processing.

2.5.4 Search the RECORD-TABLE for an entity matching RT-RTID of the record table with SS-RTID of the set spec table. When a match is found, update the entry of the RECORD-TABLE.

```
RT-FOW-FLAG     = 1
RT-VBL-FILE     = ST-MEMBER
RT-VBL-CTRL-KEY = DF-ID
```

3. Generate TOTAL RP Data Division Code.

3.1 Identification Division, Environment Division, Data Division and File Section.

Call "CDMACR" Utility with the following:

```
Library Name - TOTAL
Macro Name   - QPT01
Parameters   - DF-ID
               Processor program name
             - QPGT-FILE-NAME1
```

3.2 Generate the conceptual schema data definitions for the results file of retrieved data.

Call "CDRFT" with the following:

```
QPGT-FILE-NAME1
QPGT-SUBTRANS-ID
RFT
```

CDRFT will generate:

```
01   RESULTS-REC
     03  RES-NULL-nn    PIC 9.
     03  RES-nn   PIC type(size)[V9(nd)].

WORKING-STORAGE SECTION.

   01   CS-VAR-nn   PIC type(size)[V9(nd)].
```

4. Generate TOTAL RP Working-Storage Data Definitions.

4.1 Generate TOTAL keywords and TOTAL status values.

Call "CDMACR" utility with the following:

```
Library Name - TOTAL
Macro Name   - QPT02
Parameters   - None
File Name    - QPGT-FILE-NAME1
```

4.2 Generate data definitions required for error handling interface.

4.2.1 Generate error processing variables.

Call "CDMACR" utility with the following:

```
Library Name - TOTAL
Macro Name   - QPT03
Parameters   - qqqqq - QPGT-QP-NAME, the
                       Request Processor program
                       name.
File Name    - QPGT-FILE-NAME1
```

4.3 Generate internal schema data definitions required for the transformation of the runtime search/update values from conceptual to internal format which will be input by the user at runtime.

4.3.2 Generate the internal schema data definitions for the runtime search parameters.

Call "CDPRM" with the following

```
QPGT-FILE-NAME1
QPGT-SUBTRANS-ID
IS-QUALIFY-LIST
COMPLEX-MAPPING-ALG-TABLE
```

CDPRM will generate:

```
01  ISQL-VAR-nn              PIC type(size)
01  ISQL-mod-name-mm-nn      PIC type
                               (size)[V9(nd)].
```

4.4 Generate data definitions required for dynamically allocating the RESULTS file on an IBM system.

Call "CDMACR" utility with the following:

```
Library Name - IBM
Macro Name   - IBM01
Parameters   - none
File Name    - QPGT-FILE-NAME1
```

4.5 Generate the complete description of the TOTAL Data Areas. This area will be used to hold the fields returned by the element list.

4.5.1 Generate the 01 level entry for the record definition:

```
01  rtid-AREA.
```

where

rtid = RT-RTID

4.5.2 CALL "CDGENRT" with the following:

```
QPGT-DBID
RT-RTID
RT-RTNO
QPGT-FILE-NAME1
MODULE-STATUS
```

4.6 Generate the internal schema data definitions for the retrieved data fields and those that will be used for retrieval qualifications.

4.6.1 Call "CDQDF" with the following:

```
QPGT-FILE-NAME1
QPGT-SUBTRANS-ID
IS-QUALIFY-LIST
COMPLEX-MAPPING-ALG-TABLE
```

CDQDF will generate:

```
01  ISQL-VAR-nn          PIC type(size)
                           [V9(nd)].
01  ISQR-VAR-nn          PIC type(size)
                           [V9(nd)].
01  ISQL-mod-name-mm-nn  PIC type(size)
                           [V9(nd)].
01  ISQR-mod-name-mm-nn  PIC type(size)
                           [V9(nd)].
```

4.6.2 Call "CDRDF" with the following:

```
QPGT-FILE-NAME1
QPGT-SUBTRANS-ID
IS-ACTION-LIST
```

COMPLEX-MAPPING-ALG-TABLE

CDRDF will generate:

```
01    IS-VAR-nn     PIC type(size)
                        [V9(nd)].
01    IS-mod-name-mm-nn PIC type(size)
                              [V9(nd)].
```

4.7 Generate data definitions for TOTAL database variables.

Call "CDMACR" utility with the following:

```
Library Name - TOTAL
Macro Name   - QPT05
Parameters   - P1 - QPGT-DBMOD-NAME
          P2 - QPGT-QP-NAME
File Name    - QPGT-FILE-NAME1
```

4.8 Generate the data definitions for the TOTAL OPENX/CLOX working storage variables.

4.8.1 Generate 01 and 03 level data definitions:

```
01  REALM-OPEN.
    03  FILLER  PIC X(6) VALUE "REALM=".
```

4.8.2 For each entry in the RECORD-TABLE, use the RT-RTID of the entry and generate an 03 level data definition:

```
03  FILLER  PIC X(13) VALUE "rrSUPD****,".
```

where rr = RT-RTID(RT-INDEX)

4.8.3 Generate an 03 level data definition:

```
03  FILLER  PIC X(4) VALUE "END.".
```

4.8.4 Generate 01 and 03 level data definitions:

```
01  REALM-CLOSE.
    03  FILLER PIC X(6) VALUE "REALM=".
```

4.8.5 For each entry in the RECORD-TABLE, use the RT-RTID of the entry and generate an 03 level data definition:

```
03  FILLER  PIC X(13) VALUE "rrCOMP****,".
```

where rr = RT-RTID(RT-INDEX)

4.8.6 Generate an 03 level data definition:

```
03  FILLER  PIC X(4) VALUE "END.".
```

4.9 Generate data definitions for the data set names and the reference fields required to process the NDML subtransaction.

For each entry in the RECORD-TABLE use the RT-RTID of the entry and generate an 01 level data definition:

```
01   rt-rtid           PIC X(4) VALUE
                                `rt-rtid'.
01   rt-rtid-REFER     PIC X(5).
```

4.10 Generate the data definitions for the TOTAL linkage path variables and link fields required to procesthe NDML subtransaction.

For each entry in the GC-TABLE where GC-COMMAND = `FFM' (Find First Member):

4.10.1 Access the SET-SPEC-TABLE using the GC-SPEC-PTR from the GC-TABLE entry and extract SS-SETID.

4.10.2 Establish temporary variables for linkpaths and link fields:

```
TEMP-LINKPATH      =      SS-SETID
     (characters 1-8)

TEMP-LINKFIELD     =      SS-SETID
     (characters 5-8)
```

4.10.3 Generate 01 level data definitions:

```
01  ss-rtid-LINKPATH  PIC X(8)
                         VALUE `temp-linkpath'.
01  ss-rtid-LINK      PIC X(5)
                         VALUE `temp-linkfield'.
```

4.11 Generate the data definitions for the TOTAL data lists, and control keys required to process the NDML subtransaction.

For each entry in the RECORD-TABLE perform Steps 4.11.1 thru 4.11.4.

4.11.1 Create DATA-FIELD-TABLE entries for the control keys of current RT-RTID of the RECORD-TABLE entry.

Save the current RT-RTID.

```
TEMP-RTID = RT-RTID
```

4.11.1.1 If RT-FOW-FLAG is set (value=1), add an entry to the DATA-FIELD-TABLE if the data field to be added does not already exist in the DATA-FIELD-TABLE:

```
DF-RTID      =  TEMP-RTID
DF-DFID      =  rt-rtid CTRL
DF-DFID-TYPE =  "C"
DF-USED      =  DF-USED+1
```

4.11.1.2 For each entry in the GC-TABLE where GC-COMMAND = "FFR" (Find First Record), access the RECORD-KEY-TABLE using the GC-SPEC-PTR from the GC-TABLE entry and extract the record id (RK-RTID).

If RK-RTID does not equal TEMP-RTID, ignore the entry and continue at 4.11.1.2.

Establish a temporary data field and determine if it exists in the SEG-TABLE.

```
TEMP-DFID = rk-rtid CTRL
```

If the data field does not exist in the table, add an entry:

```
DF-RTID      =  TEMP-RTID
DF-DFID      =  TEMP-DFID
DF-DFID-TYPE =  "C"
DF-USED      =  DF-USED+1
```

If the data field does exist in the DATA-FIELD-TABLE, update the entry in the table:

```
DF-DFID-CTRL = "C"
```

4.11.2 Generate the data definitions for the TOTAL data list

4.11.2.1 Generate an 01 level data definition:

```
01   temp-rtid-LIST.
```

4.11.2.2 Access the CDM entity class DATA-FIELD to retrieve the data field name (DF-ID) with the following qualifications:

```
DB_ID              = RT-DBID
RT_ID              = TEMP-RTID
DBMS_ACCESS        = "K"
COMPONENT_OF_DF    = "NULL"

ORDER BY REC_SEQ_NO
```

If no entries are found, generate a NO-KNOWN-DATA-FIELD's user error message and terminate processing.

For each entry retrieved, use the DF-ID and generate an 03 level data definition:

```
03   FILLER   PIC X(8) VALUE "df-id".
```

4.11.2.3 Generate an 03 level data definition:

```
03   FILLER   PIC X(4) VALUE "END"
```

4.11.3 This step has been removed.

4.11.4 Generate the data definitions for the TOTAL control keys.

For each entry in the DATA-FIELD-TABLE where DF-DFID-CTRL = `C' generate an 01 level data definition:

```
01   CONTROL-df-dfid (1:4)
                              picture clause
```

Call routine "CDRTSND" to retrieve the type, size and number of decimal digits for the data field. Then call routine "CDPIC" to generate the picture clause with the return values from "CDIMD".

4.12 Generate the data definitions for complex mapping algorithm parameters.

Call "CDCMPRM" with the following:

```
QPGT-FILE-NAME1
COMPLEX-MAPPING-ARG-TABLE
QPGT-SUBTRANS-ID
MODULE-STATUS
```

CDCMPRM will generate:

```
01   PARM-mod-name-mm-nn    PIC type(size)
                                    [V9(nd)].
```

4.13 Generate the conceptual schema data definitions for the routine update/search values and the start of the Linkage Section for the request processor program.

Call "CDMSG" with the following:

```
QPGT-FILE-NAME1
QPGT-SUBTRANS-ID
CS-QUALIFY-LIST
CS-ACTION-LIST
IS-QUALIFY-LIST
IS-ACTION-LIST
```

CDMSG will generate:

```
01   CS-VAR-nn       PIC type(size)[V9(nd)].
01   CSQ-VAR-nn      PIC type(size)[V9(nd)].

LINKAGE SECTION.
```

```
01   MESSAGE-BODY-IN.
     03  CASE-NO       PIC XXX.
     03  SUBID         PIC 999.
     03  MSG-VAR-nn    PIC type(size)[V9(nd)].
     03  MSGI-VAR-nn   PIC type(size)[V9(nd)].
```

4.14 Generate data definitions for the output parameters of the request processor program.

Call "CDMACR" utility with the following:

Library Name - TOTAL
Macro Name - QPT04
Parameters - none
File Name - QPGT-FILE-NAME1

5. Generate TOTAL Request Processor Procedure Division Initialization Code.

5.1 Generate code to initialize program.

Call "CDMACR" utility with the following:

Library Name - TOTAL
Macro Name - QPT06
Parameters - none
File Name - QPGT-FILE-NAME2

5.2 Generate code to dynamically free DD name and allocate it to the RESULTS file and open RESULTS file.

Call "CDMACR" utility with the following:

Library Name - IBM
Macro Name - IBM02
Parameters - none
File Name - QPGT-FILE-NAME2

6. Generate TOTAL Request Processor Procedure Division Data Retrieval/Update code.

Process each command in the GC-TABLE and generate the appropriate code as depicted in Table 15-1.

If the GC command is one that will produce an IF statement (IF, IN1, IN2, IRF, IRN) then special processing must be performed in order to handle the possibility of nested IFs in the generated Request Processor:

- Determine if the next GC command is an `EIF'; if so, set the GC-INDEX up by one and exit out of processing for the current command.

- Stack the GC index of the current command on the IF stack

```
IF-STACK-PTR  = GC-INDEX
IF-STACK-USED = IF-STACK-USED +1
```

- Set the IF command flag to on status

  IF-COMMAND-FLAG = 1

- Continue processing with the next GC command

If the GC command is one that will not produce an IF statement and is not an 'EIF' (ENDIF) command, then determine if this command is within an IF, END-IF structure. Determine if the IF-COMMAND-FLAG is set:

- If IF-COMMAND-FLAG is set
  - Generate an IF statement for each of the IF commands on the IF stack.
  - Set the IF command flag to off status

    IF-COMMAND-FLAG = 0

  - Generate the code for the current command being processed (continue at step 6.1).
- If the IF-COMMAND-FLAG is not set
  - Generate the code for the current command being processed (continue at step 6.1).

For each entry in the GC-TABLE determine the GC command and perform the appropriate section of code as detailed below:

6.1 EIF Command

6.1.1 Generate a period (.) in the procedure division of the TOTAL Request Processor.

6.1.2 Search forward in the GC-TABLE for the next non-EIF command or end-of-table condition.

For each EIF command found, pop the IF stack one level and set the GC-INDEX down by one.

6.2 ELP Command

Generate the end of a looping construct in the procedure division of the TOTAL Request Processor.

Generate the following:

```
GO TO LOOP-nn.
   EXIT-nn.
```

6.2.1 Generate:

```
GO TO LOOP-nn.
```

If GC-SPEC-PTR equal zero, go to step 6.2.2, otherwise

```
nn = GC-SPEC-PTR
```

6.2.2 Generate:

EXIT-nn.

where:

nn = GC-SPEC-PTR

6.3 ELS Command

Generate an ELSE COBOL statement in the procedure division of the TOTAL Request Processor.

6.4 FFA Command

Generate the TOTAL commands to perform a sequential search of a data set.

6.4.1 Generate the procedure division statement to initialize the sequential search:

MOVE BEGIN TO recname-REFER.

where:

recname = RAS-RTID (GC-SPEC-PTR(GC-INDEX))

6.4.2 Generate the TOTAL call for the DML `RDNXT' command.

Call "CDMACR" utility with the following:

Library Name - TOTAL
Macro Name - RDNXT
Parameters - RECNAME - name of data set to be retrieved as determined in 6.4.1
File Name - QPGT-FILE-NAME2

6.5 FFM Command

Generate the TOTAL commands to access and retrieve a variable data set record.

6.5.1 Generate the procedure division statement to move the link field of the variable data set to the reference field:

MOVE RECNAME-LINK to recname-REFER.

where:

recname = SS-RTID(GC-SPEC-PTR(GC-INDEX))

6.5.2 Generate the TOTAL call for the DML `READV' command.

Call "CDMACR" utility with the following:

```
Library Name - TOTAL
Macro Name   - READV
Parameters   - RECNAME - name of data set to be
                        retrieved as determined
                        in 6.5.1.
File Name    - QPGT-FILE-NAME2
```

6.6 FFR Command

Generate the TOTAL commands to access a master data set record using a control key.

6.6.1 Generate the procedure division statement to move a search parameter to the control key area:

```
MOVE ISQL-VAR-nn TO CONTROL-recname.
```

where:

```
nn      = RK-ISQ-PTR (GC-SPEC-PTR(GC-INDEX)
recname = RK-RTID (GC-SPEC-PTR(GC-INDEX))
```

6.6.2 Generate the TOTAL call for the DML `READM' command.

Call "CDMACR" utility with the following:

```
Library Name - TOTAL
Macro Name   - READM
Parameters   - RECNAME - name of data set
                        to be retrieved as
                        determined in 6.6.1.
File Name    - QPGT-FILE-NAME2
```

6.7 FNA Command

Generate the TOTAL commands to retrieve the next record from a sequential search of a data set.

Generate the TOTAL call for the DML `RDNXT' command.

Call "CDMACR" utility with the following:

```
Library Name - TOTAL
Macro Name   - RDNXT
Parameters   - RECNAME - name of the data set
                        to be
                        retrieved
```

where:

```
RECNAME      = RAS-RTID(GC-SPEC-PTR(GC-INDEX))
File Name    = QPGT-FILE-NAME2
```

6.8 FNM Command

Generate the TOTAL commands to retrieve the next record in a variable data set along a linkpath.

Generate the TOTAL call for the DML `READV' command.

Call "CDMACR" utility with the following:

```
Library Name - TOTAL
Macro Name   - READV
Parameters   - RECNAME - of the data set to be
                        retrieved
```

where:

```
RECNAME      = SS-RTID(GC-SPEC-PTR(GC-INDEX))
File Name    = QPGT-FILE-NAME2
```

6.9 FNR Command

Generate the following statement in the procedure division

```
MOVE `END.' TO REFER.
```

NOTE: FNR is not defined to TOTAL since duplicates are not allowed. Hence FNR will set up the REFER to keep logic consistent and generate a meta data error.

6.10 FOW Command

Generate the TOTAL commands to access a master data set record.

6.10.1 Generate the procedure division statement to move the variable control key of the variable data set to the control key of the master data set.

6.10.1.1 Access the SET-SPEC-TABLE using the GC-SPEC-PTR from the GC-TABLE entry and extract SS-SETID and SS-RTID.

6.10.1.2 Search the SET-TABLE for an entry matching current database id (QPGT-DBID), current subtrans id (QPGT-SUBTRANS), SS-SETID and SS-RTID saved in 6.10.1.1 with the ST-DBID, ST-SUBTRANS-ID, ST-SETID and ST-OWNER of the SET-TABLE entry. When a match is found, extract the record id of the member occurrence (ST-MEMBER).

6.10.1.3 Search the RECORD-TABLE for an entry matching SS-RTID with RT-RTID of the record table entry and ST-MEMBER with RT-VBL-FILE of the record table entry. When a match is found, extract the variable control key (RT-VBL-CTRL-KEY).

6.10.1.4 Generate the procedure division statement:

MOVE data-field to CONTROL-recname

where:

data-field = RT-VBL-CTR-KEY from step 6.10.1.3
recname = SS-RTID (GC-SPEC-PTR(GC-INDEX))

6.10.2 Generate the TOTAL call for the DML `READM' command.

Call "CDMACR" utility with the following:

Library Name - TOTAL
Macro Name - READM
Parameters - recname name of the data set to be retrieved as determined in 6.10.1.4.
File Name - QPGT-FILE-NAME2

6.11 GI1 Command

This command requires no code to be generated since each TOTAL DML function READM, READV and RDNXT transfers data to working storage.

6.12 GI2 Command

This command requires no code to be generated since each TOTAL DML function READM, READV and RDNXT transfers data to working storage.

6.13 GIF Command

Generate code to determine if a retrieved data field contains a null value and establish the null flag for this result field. If the retrived field does not contain a null value, generate code to move the retrieved data field which is in internal format to the results file which is in conceptual format. Generate numeric testing of retrieved data before moving it to the results file to avoid abnormal end of processing.

6.13.1 Generate the following procedure division code if the retrieved data field has a character data type
(RF1-TYPE(GC-SPEC-PTR(GC-INPCX)) = "C"):

IF D-dfno = character-null-value
MOVE 1 TO RES-NULL-nn
MOVE ZEROES TO RES-nn

```
IF D-dfno NOT = null-value
    MOVE 0 TO RES-NULL-nn
    MOVE D-dfno TO RES-nn.
```

where:

dfno = RF1-DFNO(GC-SPEC-PTR(GC-INDEX))
character-null-value = CHARACTER-NULL-VALUE from input parameter

nn = RF1-IS-PTR(GC-SPEC-PTR(GC-INDEX))

6.13.2 Generate the following procedure division code if the retrieved data field has a numeric data type (RF1-TYPE(GC-SPEC-PTR(GC-INDEX))NOT = "C"):

```
IF D-dfno NOT NUMERIC
   STRING "DATA CONTAINED IN DATA FIELD"
   RF1-DFNO (GC-SPEC-PTR(GC-INDEX))
            " FOR RECORD "
   RF1-RTID (GC-SPEC-PTR(GC-INDEX))
            " IS NON NUMERIC "
   DELIMITED BY SIZE
   INTO MESG-DESC
   PERFORM PROCESS-ERROR
       MOVE 1 TO RES-NULL-nn
       MOVE ZEROES TO RES-nn
   ELSE
IF D-dfno = numeric-null-value
       MOVE 1 TO RES-NULL-nn
       MOVE ZEROES TO RES-nn
   ELSE
       MOVE 0 TO RES-NULL-nn
       MOVE D-dfno TO RES-nn
```

where:

dfno = RF1-DFNO(GC-SPEC-PTR(GC-INDEX))
numeric-null-value = NUMERIC-NULL-VALUE from input parameter
nn = RF1-IS-PTR(GC-SPEC-PTR(GC-INDEX))

6.14 GIO Command

Generate code to determine if a retrieved data field contains a null value and establish the contents of a local variable and its null flag declared in the working storage section. Generate code to perform numeric checks on retrieved data before moving it to a local variable to avoid abnormal end of processing.

Generate the following if FG1-TYPE(GC-SPEC-PTR(GC-INDEX)) = "C":

```
IF D-dfno = null-value
   MOVE 1 TO ISQside-NULL-nn
   MOVE ZEROES TO ISQside-VAR-nn
```

```
 IF  D-dfno NOT = null-value
     MOVE 0 TO ISQside-NULL-nn
     MOVE D-dfno TO ISQside-VAR-nn

Generate the following if
FG1-TYPE(GC-SPEC-PTR(GC-INDEX))  NOT = "C":

IF D-dfno NOT NUMERIC
   STRING "DATA CONTAINED IN DATA FIELD"
           FG1-DFNO (GC-SPEC-PTR(GC-INDEX))
              " FOR RECORD "
           FG1-RTID (GC-SPEC-PTR(GC-INDEX))
              " IS NON NUMERIC "
   DELIMITED BY SIZE
   INTO MESG-DESC
     PERFORM PROCESS-ERROR
        MOVE 1 TO ISQside-NULL-nn
        MOVE ZEROES TO ISQ-side-VAR-nn
  ELSE
    IF D-dfno = null-value
        MOVE 1 TO ISQside-NULL-nn
        MOVE ZEROES TO ISQside-VAR-nn
 ELSE
        MOVE 0 TO ISQside-NULL-nn
        MOVE D-dfno TO ISQside-VAR-nn

  where:

  dfno       = FG1-DFNO (GC-SPEC-PTR(GC-INDEX))
  side       = FG1-SIDE (GC-SPEC-PTR(GC-INDEX))
  nn         = FG1-ISQ-PTR (GC-SPEC-PTR(GC-INDEX))
  null-value = CHARACTER-NULL-VALUE from input
                                        parameter
               if:
                FG1-TYPE(GC-SPEC-PTR(GC-INDEX)) = "C"
             = NUMERIC-NULL-VALUE from input parameter
               if:
                FG1-TYPE(GC-SPEC-PTR(GC-INDEX))NOT
                                              ="C"
```

6.15 IF Command

Generate code for an IF condition check of a value and a variable holding a retrieved data field.

Generate the following procedure division statement:

IF (value op VARIABLE-nn)

where:

```
value     = IT2-VALUE (GC-SPEC-PTR(GC-INDEX))
op        = IT2-OP (GC-SPEC-PTR(GC-INDEX))
nn        = IT2-ISQ-PTR (GC-SPEC-PTR(GC-INDEX))
variable  = "IS-VAR" if
                IT2-ISQ-PTR(GC-SPEC-PTR(GC-INDEX)) = 0
          = "ISQL-VAR" if
            IT2-ISQ-PTR(GC-SPEC-PTR(GC-INDEX)) NOT = 0
```

6.16 IN1 Command

Generate code for an IF NOT condition of two retrieved data fields.

Generate the following procedure division statement:

```
IF NOT  (D-dfno1 = null-value1 AND
         D-dfno2 = null-value2)
AND NOT (D-dfno1 op D-dfno2)
```

where:

```
null-value1 = CHARACTER-NULL-VALUE from input parameter if:
              RS1-TYPE1(GC-SPEC-PTR(GC-INDEX)) = "C"
            = NUMERIC-NULL-VALUE from input parameter if:
              RS1-TYPE1(GC-SPEC-PTR(GC-INDEX)) NOT = "C"
null-value2 = CHARACTER=NULL-VALUE from input parameter if:
              RS2-TYPE2(GC-SPEC-PTR(GC-INDEX)) = "C
            = NUMERIC-NULL-VALUE from input parameter if:
              RS2-TYPE2(GC-SPEC-PTR(GC-INDEX)) NOT = "C"
dfno1       = RS1-DFNOL (GC-SPEC-PTR(GC-INDEX))
op          = RS1-OP    (GC-SPEC-PTR(GC-INDEX))
dfno2       = RE1-DFNOR (GC-SPEC-PTR-(GC-INDEX))
```

6.17 IN2 Command

Generate code for an IF NOT condition of a data field and a variable holding a retrieved data field.

Generate one of the following procedure division statements.

If RS4-SIDE (GC-INDEX) = "L", generate:

```
IF NOT (D-dfno = null-value)
AND NOT (D-dfno op ISQL-VAR-nn)
```

If RS4-SIDE (GC-INDEX) = "R", generate:

```
IF NOT (D-dfno = null-value)
AND NOT (D-dfno op ISQR-VAR-nn)
```

where:

```
dfno       = RS4-DFNO (GC-SPEC-PTR(GC-INDEX))
op         = RS4-OP   (GC-SPEC-PTR(GC-INDEX))
nn         = RS4-ISQ-PTR (GC-SPEC-PTR(GC-INDEX))
null-value = CHARACTER-NULL-VALUE from input
                                     parameter if:
             RS4-TYPE(GC-SPEC-PTR(GC-INDEX)) = "C"
                    = NUMERIC-NULL-VALUE from input
                                     parameter if:
           RS4-TYPE (GC-SPEC-PTR(GC-INDEX)) not = "C"
```

6.18 IRF Command

Generate code that will determine if the last TOTAL DML call was successful in retrieving a record.

Generate the following procedure division statement:

IF (SUCCESSFUL AND NOT END-OF-RECORDS)

6.19 IRN Command

Generate code that will determine if the last TOTAL DML call did not retrieve a record.

Generate the following procedure division statement:

IF (END-OF-RECORDS AND SUCCESSFUL)

6.20 LOP Command

Generate a paragraph name required for the retrieval loop of the Request Processor.

Generate the following procedure division statement:

LOOP-nn.

where:

nn = GC-SPEC-PTR(GC-INDEX)

6.21 NLP Command

Generate code to transfer control to the current retrieval loop paragraph for the next record.

Generate the following procedure division statement:

GO TO EXIT-nn.

where:

nn = GC-SPEC-PTR(GC-INDEX) +1

6.22 PIO Command

Generate code that will write retrieved data to the results file and increment the retrieved record count.

Generate the WRITE statement.

Call "CDMACR" utility with the following:

Library Name - TOTAL
Macro Name - PIO
Parameters - GC-SPEC-PTR(GC-INDEX) + 1

6.23 RCR Command

Generate code that will reset currency for the current record.

Generate the following procedure division statement:

```
MOVE SPACE TO REFER.
```

6.24 XLP Command

Generate code to transfer control to the current exit paragraph.

Generate the following procedure division statement:

```
GO TO EXIT-nn
```

where:

nn = GC-SPEC-PTR(GC-INDEX)

6.25 This step has been replaced with paragraph 6.49.

6.26 RIK Command

Generate the TOTAL commands to add a new record to a master (owner) file.

6.26.1 Generate the procedure division statement to move a key value to the control key area:

MOVE keyvalue TO CONTROL-recname.

where:

keyvalue = RI1-KEYVALUE (GC-SPEC-PTR(GC-INDEX))
recname = RI1-RTID (GC-SPEC-PTR(GC-INDEX))

6.26.2 Generate the TOTAL call for the DML `ADD-M' command.

Call "CDMACR" utility with the following:

Library Name - TOTAL
Macro Name - ADD-M
Parameters - recname (as defined in 6.26.1)
File Name - QPGT-FILE-NAME2

6.27 RDK Command

Generate the TOTAL commands to delete a record from a master (owner) file.

Generate the TOTAL call for the DML `DEL-M' command.

Call "CDMACR" utility with the following:

```
Library Name - TOTAL
Macro Name   - DEL-M
Parameters   - RECNAME = RD1-RTID
               (GC-SPEC-PTR(GC-INDEX))
File Name    - QPGT-FILE-NAME2
```

6.28 RUK Command

Generate the TOTAL commands to update in place an existing record in a master (owner) file.

Generate the TOTAL call for the DML `WRITM' command.

Call "CDMACR" utility with the following:

```
Library Name - TOTAL
Macro Name   - WRITM
Parameters   - RECNAME = RU1-RTID
               (GC-SPEC-PTR(GC-INDEX))
File Name    - QPGT-FILE-NAME2
```

6.29 RIS Command

Generate the TOTAL commands to add a new variable (member) record to the end of all associated chains.

Generate the TOTAL call for the DML `ADDVC' command.

Call "CDMACR" utility with the following:

```
Library Name - TOTAL
Macro Name   - ADDVC
Parameters   - RECNAME = RI2-RTID
               (GC-SPEC-PTR(GC-INDEX))
File Name    - QPGT-FILE-NAME2
```

6.30 RDS Command

Generate the TOTAL commands to delete a variable (member) record.

Generate the TOTAL call for the DML `DELVD' command.

Call "CDMACR" utility with the following:

```
Library Name - TOTAL
Macro Name   - DELVD
Parameters   - RECNAME = RD2-RTID
               (GC-SPEC-PTR(GC-INDEX))
File Name    - QPGT-FILE-NAME2
```

6.31 RUS Command

Generate the TOTAL commands to update in place an existing variable (member) record.

Generate the TOTAL call for the DML `WRITV' command.

Call "CDMACR" utility with the following:

```
Library Name - TOTAL
Macro Name   - WRITV
Parameters   - RECNAME = RU2-RTID
     (GC-SPEC-PTR(GC-INDEX))
File Name    - QPGT-FILE-NAME2
```

6.32 COMMIT Command

The Request Processor Main program will handle the COMMIT function.

6.33 ROLLBACK Command

The Request Processor Main program will handle the ROLLBACK function.

6.34 CIF Command

Generate an IF statement to evaluate the qualifications in the NDML WHERE clause at the conceptual schema level.

6.35 CAL Command

Generate code in the procedure division of the TOTAL Request Processor to call a complex mapping algorithm for the transformation of data.

6.35.1 Generate code to bypass the call to the complex mapping algorithm if any null values:

```
IF PARM-NULL-alg-modinst = 1
GO TO alg-modinst.
```

where:

```
alg     = CAL-ALG-ID(GC-SPEC-PTR(GC-INDEX))
modinst = CAL-MOD-INST(GC-SPEC-PTR(GC-INDEX))
```

6.35.2 Generate code to call the complex mapping algorithm:

```
CALL "alg"  USING PARM-alg-modinst-parmno
                  .
                  .
                  .
                  .
```

where:

```
alg     = CAL-ALG-ID(GC-SPEC-PTR(GC-INDEX))
modinst = CAL-MOD-INST(GC-SPEC-PTR(GC-INDEX))
parmno  = 1 - n
n       = CAL-PARM-COUNT(GC-SPEC-PTR(GC-INDEX))
```

6.35.3 Generate standard error handling code to process the return status from the complex mapping algorithm.

6.35.4 Generate the bypass point for the call to the complex mapping algorithm:

alg-modinst.

where:

```
alg      = CAL-ALG-ID(GC-SPEC-PTR(GC-INDEX))
MODINST  = CAL-MOD-INST(GC-SPEC-PTR(GC-INDEX))
```

6.36 EP Command

Generate the end of a loop construct of the TOTAL Request Processor:

```
EXIT-nn.
```

where:

```
nn = GC-SPEC-PTR + 1
```

6.37 GF2 Command

Generate code to move a set value to a local variable declared in the working storage section:

```
MOVE `set-value' TO ISQside-VAR-nn
```

where:

```
set-value  = FG2-DFID(GC-SPEC-PTR(GC-INDEX))
side       = FC2-SIDE(GC-SPEC-PTR(GC-INDEX))
nn         = FG2-ISQ-PTR(GC-SPEC-PTR(GC-INDEX))
```

6.38 IFC Command

Generate code for a condition of a data field and local variable in an IF statement.

Generate the following procedure division statement:

```
condition ((D-dfno NOT = null-value AND
           ISQside-NULL-nn NOT = 1) AND
           (D-dfno op ISQside-VAR-nn))
```

where:

```
condition:  =  RS5-IF-OR (GC-SPEC-PTR(GC-INDEX))
dfno        =  RS5-DFNO
side        =  RS5-SIDE
op          =  RS5-OP
null-value  =  CHARACTER-NULL-VALUE from input
                                        parameter
if:
RS5-TYPE (GC-SPEC-PTR(GC-INDEX)) = "C"
        = NUMERIC-NULL-VALUE from input parameter.
if:
```

RS5-TYPE (GC-SPEC-PTR(GC-INDEX)) NOT = "C"

6.39 IFX Command

Generate code to determine if the current index is greater than the maximum index value.

Generate the following procedure division statement:

```
IF INDEX-dfno > INDEX-dfno-MAX
```

where:

dfno = OC4-INDEX-DFNO(GC-SPEC-PTR(GC-INDEX))

6.40 IF1 Command

Generate code for the following conditional test of a looping construct for key values:

```
ADD 1 TO LOOP-COUNT
   IF LOOP-COUNT > max
        GO TO EXIT-(n)
```

where:

max = RK1-LOOP-MAX (GC-SPEC-PTR(GC-INDEX))
nn = GC-SPEC-PTR (GC-INDEX) - 1

6.41 IF2 Command

Generate code for the following conditional test of looping construct for repeating data fields:

```
IF LOOP-COUNT = n
```

where:

n = RK2-LOOP-COUNT (GC-SPEC-PTR(GC-INDEX))

6.42 IIF Command

Generate an IF statement to evaluate the qualifications in the NDML WHERE clause at the internal schema level.

6.43 IN3 Command

Generate code for an IF NOT condition of a set value and a variable holding retrieved data field.

Generate the following procedure division statement:

```
IF NOT ("set-value" op ISQside-VAR-nn)
```

where:

```
set-value = RS3-DFID(GC-SPEC-PTR(GC-INDEX))
op        = RS3-OP  (GC-SPEC-PTR(GC-INDEX))
side      = RS3-SIDE(GC-SPEC-PTR(GC-INDEX))
nn        = RSE-ISQ-PTR
```

6.44 This step was removed.

6.45 This step was removed.

6.46 This step was removed.

6.47 MR1 Command

This command requires no code to be generated since each TOTAL DML function transfers data to working storage.

6.48 MR2 Command

This command requires no code to be generated since each TOTAL DML function transfers data from working storage.

6.49 MV Command

Generate code to move a conceptual schema update value to an internal schema data field. If this field has not been mapped from the entity class being updated, null values will be moved to the data field.

IF FUS-NULL (GC-SPEC-PTR(GC-INDEX)) = 0 generate:

```
MOVE MSG-VARI-nn TO D-dfno
```

where:

```
nn   = FUS-IS-PTR(GC-SPEC-PTR(GC-INDEX))
dfno = FUS-DFNO(GC-SPEC-PTR(GC-INDEX)) NOT = 0
```

IF FUS-NULL(GC-SPEC-PTR(GC-INDEX)) NOT = 0 generate:

```
MOVE null-value TO D-dfno
```

where:

```
dfno       = FUS-DFNO(GC-SPEC-PTR(GC-INDEX))
null-value = CHARACTER-NULL-VALUE from input
parameter if:
        FUS-TYPE (GC-SPEC-PTR(GC-INDEX)) = "C"
               = NUMERIC-NULL-VALUE from input
                                    parameter if:
        FUS-TYPE (GC-SPEC-PTR(GC-INDEX)) NOT = "C"
```

6.50 MVI Command

Generate code to move variable containing the number of occurrences or occurs depending on value to a local variable.

Generate:

```
MOVE ISQL-VAR-nn TO INDEX-dfno-MAX
```

where:

```
nn   = OC2-ISQ-PTR (GC-SPEC-PTR(GC-INDEX))
dfno = OC2-INDEX-DFNO
```

6.51 MVK Command

Generate code to move a key field value to a key field of a record.

```
MOVE ISQL-VAR-nn TO D-dfno
```

where:

```
nn   = RK2-RK-INDEX (GC-SPEC-PTR(GC-INDEX))
dfno = RK2-DFNO (GC-SPEC-PTR(GC-INDEX))
```

6.52 MVM Command

Generate code to move data field or value containing the number of occurrences or occurs depending on value to a local variable:

IF OC3-MAX-OCCURS (GC-SPEC-PTR(GC-INDEX)) > 0 generate:

```
MOVE nn TO INDEX-indexdfno - MAX
```

otherwise generate:

```
MOVE D-dfno TO INDEX-indexdfno-MAX.
```

where:

```
nn        = OC3-MAX-OCCURS (GC-SPEC-PTR(GC-INDEX))
indexdfno = OC3-INDEX-DFNO (GC-SPEC-PTR(GC-INDEX))
dfno      = OC3-OCCURS-DEP-DFNO
(GC-SPEC-PTR(GC-INDEX))
```

6.53 MVS Command

Generate code to move a conceptual schema search value to a local variable in internal schema format.

```
MOVE MSG-VAR-nn TO ISQL-VAR-nn
```

where:

```
nn = MVS-ISQ-PTR (GC-SPEC-PTR(GC-INDEX))
```

6.54 MVX Command

Generate code to move an indexed field to the results record:

```
MOVE D-dfno (INDEX-dfno1,INDEX-dfno2,INDEX-dfno3) TO
RES-nn
```

where:

```
dfno  = OC5-DFNO   (GC-SPEC-PTR(GC-INDEX))
nn    = OC5-IS-PTR
```

dfno1, dfno2, dfno3 are used depending on the value of OC5-NUM-INDEXES and are found in OC5-IDX-DFN, OC5-IDX-DFNO2, OC5-IDX-DFNO3

6.55 MV2 Command

Generate code to initialize the local variable controlling the number of iterations through the construct for multiple values of a key.

```
MOVE ZERO TO LOOP-COUNT.
```

6.56 MV3 Command

Generate code to move a conceptual schema update value to the input parameter of a complex mapping algorithm. Generate code to initialize the null flag for the algorithm.

```
MOVE MSG-VARI-nn TO
PARM-alg-modinst-parmno.
MOVE ZERO TO PARM-NULL-alg-modinst.
```

where:

```
nn      = FG3-IS-PTR    (GC-SPEC-PTR(GC-INDEX))
alg     = FG3-ALG-ID    (GC-SPEC-PTR(GC-INDEX))
modinst = FG3-MOD-INST  (GC-SPEC-PTR(GC-INDEX))
parmno  = FG3-PARM-NO   (GC-SPEC-PTR(GC-INDEX))
```

6.57 MV4 Command

Generate code to move a constant value to an input parameter of a complex mapping algorithm.

```
MOVE constant-value TO PARM-alg-modinst-parmno
```

where:

```
constant-value = FG4-CMA-CONSTANT (GC-SPEC-PTR(GC-INDEX))
alg            = FG4-ALG-ID       (GC-SPEC-PTR(GC-INDEX))
modinst        = FG4-MOD-INST     (GC-SPEC-PTR(GC-INDEX))
parmno         = FG4-PARM-NO      (GC-SPEC-PTR(GC-INDEX))
```

6.58 MV5 Command

Generate code to move an output parameter of a complex mapping algorithm to a working storage record description structure. Generate code to set the null flag indicator for this algorithm to zero.

```
MOVE PARM-alg-modinst-oarmno TO rtid-AREA.

MOVE ZERO TO PARM-NULL-alg-modinst.
```

where:

```
alg      = FU2-ALG-ID    (GC-SPEC-PTR(GC-INDEX))
modinst  = FU2-MOD-INST  (GC-SPEC-PTR(GC-INDEX))
parmno   = FU2-PARM-NO   (GC-SPEC-PTR(GC-INDEX))
rtid     = FU2-RTID      (GC-SPEC-PTR(GC-INDEX))
```

6.59 MV6 Command

Generate code to move an output parameter of a complex mapping algorithm to a data field. Generate code to set the null flag indicator for this algorithm to zero.

```
MOVE PARM-alg-modinst-parmno TO D-dfno.
MOVE ZERO TO PARM-NULL-alg-modinst.
```

where:

```
alg      = FU1-ALG-ID   (GC-SPEC-PTR(GC-INDEX))
modinst  = FU1-MOD-INST(GC-SPEC-PTR(GC-INDEX))
parmno   = FU1-PARM-NO  (GC-SPEC-PTR(GC-INDEX))
dfno     = FU1-DFNO     (GC-SPEC-PTR(GC-INDEX))
```

6.60 MV7 Command

Generate code to move a non-null data field to an input parameter of a complex mapping algorithm. Generate code to set the null flag indicator for this algorithm based on the data field value. Generate code to perform numeric checks on retrieved data fields to avoid an abnormal end of processing.

Generate the following if
FU3-TYPE(GC-SPEC-PTR(GC-INDEX)) = "C"

```
IF D-dfno = null-value
   MOVE 1 TO PARM-NULL-alg-modinst
   MOVE ZEROES TO PARM-alg-modinst-parmno

IF D-dfno NOT = null-value
    MOVE 0 TO PARM-NULL-alg-modinst
    MOVE D-dfno TO PARM-alg-modinst-parmno
```

Generate the following if
FU3-TYPE(GC-SPEC-PTR(GC-INDEX)) NOT = "C"

```
IF D-dfno NOT NUMERIC
    STRING "DATA CONTAINED IN DATA FIELD "
   FU3-DFNO (GC-SPEC-PTR(GC-INDEX))
      " FOR RECORD "
      FU3-RTID(GC-SPEC-PTR(GC-INDEX))
      " IS NON NUMERIC"
   DELIMITED BY SIZE
```

```
   INTO MESG-DESC
   PERFORM PROCESS-ERROR
    MOVE 1 TO PARM-NULL-alg-modinst-parmno
    MOVE ZEROES TO PARM-alg-modinst-parmno
ELSE
IF D-dfno = null-value
    MOVE 1 TO PARM-NULL-alg-modinst
    MOVE ZEROES TO PARM-alg-modinst-parmno
ELSE
    MOVE 0 TO PARM-NULL-alg-modinst
    MOVE D-dfno TO PARM-alg-modinst-parmno
```

where:

```
dfno       = FU3-DFNO     (GC-SPEC-PTR(GC-INDEX))
alg        = FU3-ALG-ID   (GC-SPEC-PTR(GC-INDEX))
modinst    = FU3-MOD-INST (GC-SPEC-PTR(GC-INDEX))
null-value = CHARACTER-NULL-VALUE from input
parameter if:
            FU3-TYPE (GC-SPEC-PTR(GC-INDEX)) = "C"
           = NUMERIC-NULL-VALUE from input parameter
                                                 if:
          FU3-TYPE (GC-SPEC-PTR(GC-INDEX)) NOT = "C"
```

6.61 MV8 Command

Generate code to move the contents of a record to an input parameter of a complex mapping algorithm. Generate code to set the null flag indicator for the algorithm to zero.

```
    MOVE rtid-AREA TO PARM-alg-modinst-parmno.
    MOVE 0 TO PARM-NULL-alg-modinst.
```

where:

```
    rtid    = FU4-RTID     (GC-SPEC-PTR(GC-INDEX))
    alg     = FU4-ALG-ID   (GC-SPEC-PTR(GC-INDEX))
    modinst = FU4-MOD-INST (GC-SPEC-PTR(GC-INDEX))
    parmno  = FU4-PARM-NO  (GC-SPEC-PTR(GC-INDEX))
```

6.62 NXS Command

Generate a COBOL "NEXT SENTENCE" statement in the procedure division of the Request Processor.

```
    NEXT SENTENCE
```

6.63 OU1 Command

This command requires no code to be generated.

6.64 OU2 Command

Generate code to move a set value to the results record.

```
    MOVE set-value TO RES-nn
```

where:

```
set-value = RF2-VALUE (GC-SPEC-PTR(GC-INDEX))
nn        = RF2-PTR   (GC-SPEC-PTR(GC-INDEX))
```

6.65 OU3 Command

Generate code to move the output parameter of a complex algorithm to the results record and establish the null flag for this result field.

```
IF PARM-NULL-alg-modinst = 1
   MOVE 1 TO RES-NULL-nn
   MOVE ZEROS TO RES-nn

IF PARM-NULL-alg-modinst = 0
   MOVE 0 TO RES-NULL-nn
   MOVE PARM-alg-modinst-parmno TO RES-nn
```

where:

```
alg      = RF3-ALG-ID                (GC-SPEC-PTR(GC-INDEX))
modinst  = RF3-MOD-INST              (GC-SPEC-PTR(GC-INDEX))
nn       = IS-RFT-PTR(RF3-IS-PTR     (GC-SPEC-PTR(GC-INDEX))
parmno   = RF3-PARM-NO(GC-SPEC-PTR   (GC-INDEX))
```

6.66 OU4 Command

Generate code to move the output parameter of a complex mapping algorithm to a local variable and establish the null flag for this variable.

```
IF PARM-NULL-alg-modinst = 1
   MOVE 1 TO TAG-NULL-tagno
   MOVE ZEROES TO TAG-tagno

IF PARM-NULL-alg-modinst = 0
   MOVE 0 TO TAG-NULL-tagno
   MOVE PARM-alg-modinst-parmno TO TAG-tagno
```

where:

```
alg      = RF3-ALG-ID    (GC-SPEC-PTR(GC-INDEX))
modinst  = RF3-MOD-INST  (GC-SPEC-PTR(GC-INDEX))
tagno    = RF3-TAGNO     (GC-SPEC-PTR(GC-INDEX))
parmno   = RF3-PARM-NO   (GC-SPEC-PTR(GC-INDEX))
```

6.67 OU5 Command

Generate code to move a non-null data field to a local variable and establish the null flag for the variable. Generate code to perform numeric checks on retrieved data to avoid an abnormal end of processing.

Generate the following if OU5-TYPE(GC-SPEC-PTR(GC-INDEX)) = "C":

```
IF D-dfno = null-value
   MOVE 1 TO TAG-NULL-tagno
   MOVE ZEROES TO TAG-tagno

IF D-dfno NOT = null-tagno
   MOVE 0 TO TAG-NULL-tagno
   MOVE D-dfno TO TAG-tagno

Generate the following if
OU5-TYPE(GC-SPEC-PTR(GC-INDEX))  NOT = "C":

IF D-dfno NOT NUMERIC
    STRING "DATA CONTAINED IN DATA FIELD "
    OU5-DFNO(GC-SPEC-PTR(GC-INDEX))
    " FOR RECORD "
            OU5-RTID(GC-SPEC-PTR(GC-INDEX))
    " IS NON NUMERIC"
            DELIMITED BY SIZE
            INTO MESG-DESC
            PERFORM PROCESS-ERROR
    MOVE 1 TO TAG-NULL-tagno
    MOVE ZEROES TO TAG-tagno
ELSE
IF D-dfno = null-value
    MOVE 1 TO TAG-NULL-tagno
    MOVE ZEROES TO TAG-tagno
ELSE
    MOVE 0 TO TAG-NULL-tagno
    MOVE D-dfno TO TAG-tagno
```

where:

```
dfno       = OU5-DFNO (GC-SPEC-PTR(GC-INDEX))
tagno      = OU5-TAGNO  (GC-SPEC-PTR(GC-INDEX))
null-value = CHARACTER-NULL-VALUE from input
parameter if:
           OU5-TYPE (GC-SPEC-PTR(GC-INDEX)) = "C"
           = NUMERIC-NULL-VALUE from input parameter
if:
         OU5-TYPE (GC-SPEC-PTR(GC-INDEX)) NOT = "C"
```

6.68 SXI Command

Generate code to establish the value of an index field from a local variable.

```
    SET INDEX-dfno TO ISQL-VAR-nn
```

where:

```
    dfno = OC2-INDEX-DFNO (GC-SPEC-PTR(GC-INDEX))
    nn   = OC2-ISQ-PTR    (GC-SPEC-PTR(GC-INDEX))
```

6.69 SX1 Command

Generate code to initialize an index field to value of 1.

```
       SET INDEX-dfno TO 1.
```

where:

dfno = OC1-INDEX-DFNO (GC-SPEC-PTR(GC-INDEX))

6.70 UIF Command

Call "CDRUIF" using:

SUBTRANS-ID
SUBTRANS-BOOLEAN-LIST
WORKFILE
RECORD-TYPE-NUMBER

7. Generate TOTAL Request Processor Procedure Division Termination

7.1 Generate code to process errors and send results/status to the RP main program. Call "CDMACR" utility with the following:

Library Name - TOTAL
Macro Name - QPT09
Parameters - none
File Name - QPGT-FILE-NAME2

8. Combine the two sequential work files used in generating the Request Processor.

Call "CDCWF" with the following:

QPGT-FILE-NAME1
QPGT-FILE-NAME2

9. Return to PRE13 to continue generating code to satisfy the NDML request.

25.3 Outputs

1. Name of the file containing the request processor program.

```
01   RP-NAME                PIC X(30).
```

2. Function status.

```
01   RET-STATUS             PIC X(5).
```

3. TOTAL Request Processor program source code.

The Data Division of each TOTAL Request Processor consists of:

a) FILE SECTION containing the file description of the output RESULT-REC.

b) WORKING-STORAGE section containing the data definitions of TOTAL keywords, status values, data set names, data lists, data areas, control keys, linkage paths and link fields. It will also contain data definitions of the NDML/Update parameters.

The Procedure Division of each TOTAL Request Processor consists of:

a) Code to dynamically allocate the results file.

b) Conceptual to internal data transform on runtime search/update parameters.

c) Retrieval loop containing TOTAL DML calls to access and retrieve data from the TOTAL database. The code for these TOTAL calls will be generated using the generic DML commands found in the GC-TABLE. A cross reference of the code for the TOTAL calls that are generated from the generic DML commands can be found in Table 15-1.

d) Status code checks and error processing routines.

e) Output parameters with completion status message for the Request Processor main program.

f) Standard Request Processor error handling procedures.

25.4 Internal Requirements

1. Record Table

The Record Table contains a list of all data sets in the TOTAL database that will be accessed to satisfy a single NDML subtransaction.

```
01  RECORD-TABLE
    03  RT-MAX                      PIC 99 VALUE 25.
    03  RT-USED                     PIC 99 VALUE 0.
    03  RT-DBID                     PIC 9(6).
    03  RT-ENTRY OCCURS 25 INDEXED BY RT-INDEX.
        05  RT-RTID                 PIC X(4).
        05  RT-FOW-FLAG             PIC 9.
        05  RT-VBL-FILE             PIC X(4).
        05  RT-VBL-CTRL-KEY         PIC X(8).
        05  RT-RTNO                 PIC 9(6).
```

2. Data Field Table

The Data Field Table contains a list of all TOTAL element names for a particular data set. This table will be built and used when generating the working-storage definitions for data set lists and data set areas.

```
01  DATA-FIELD-TABLE.
    03  DF-MAX                      PIC 99 VALUE 25.
    03  DF-USED                     PIC 99 VALUE 0.
    03  DF-ENTRY OCCURS 25 TIMES
        INDEXED BY DF-INDEX.
        05  DF-DFID                 PIC X(8).
        05  DF-SOURCE-TABLE         PIC 9.
            88 IS-ACTION-TABLE      VALUE 1.
        05  DF-TABLE-PTR            PIC 99.
        05  DF-DFID-CTRL            PIC X.
            88 CONTROL-KEY          VALUE `C'.
        05  DF-DFID-TYPE            PIC X.
        05  DF-DFID-SIZE            PIC 9(3).
        05  DF-DFID-ND              PIC 9(2).
```

3. IF Stack Table

   The IF Stack Table will be used to stack all IF commands that must be processed when nested IF statements are required in the Request Processor.

```
01  IF-STACK-TABLE.
    03  IF-STACK-MAX          PIC 99  VALUE 10.
    03  IF-STACK-USED         PIC 99  VALUE ZERO.
    03  IF-STACK OCCURS 10 INDEXED BY IF-INDEX.
        05  IF-STACK-PTR      PIC 999.
01  IF-COMMAND-FLAG           PIC 9.
    88 IF-COMMAND-TO-BE-WRITTEN      VALUE 1.
```

25.5 Constraints

1. One-to-one relationships between a master data set and a variable data set, the master being used as a simple index into the variable, will not be supported. This structure (one-to-one mappings via symbolic calc keys or actual data base keys) is possible in CODASYL and is not currently supported for IISS CODASYL database management systems. This would involve mapping a single entity class to two record types. While this mapping is possible in the existing CDM1 model (RT join structures) no algorithms currently exist that use this information in the same subtransaction. Two subtransactions would result with results being joined by the CDM runtime architecture.

2. Implied relationships such as one-to-one or one-to-many among coded records in a variable file based on physical position in the chain will not be supported since this is currently not supported for IISS CODASYL database systems.

```
CIF       CIF      1. IF statement for WHERE clause
                      qualification (conceptual schema)

CAL       CAL      1.  IF PARM-NULL-alg-modinst = 1
                           GO TO alg-modinst
                   2.  CALL "alg-id" USING
                      PARM-alg-modinst-parmno
                             .
                             .
                             .
                             .
                   3.  error handling
                   4.  alg-modinst

EIF       N/A      1.  . (period)

ELP       NN       1.  GO TO LOOP-nn.
                   2.  EXIT-nn.

ELS       N/A      1.  ELSE

EP         -       1.  EXIT-nn. (+1)

FFA       RAS        1.  MOVE BEGIN TO recname-REFER.
                     2.  CALL "DATBAS" USING RDNXT,
                                             STATUS,
                                             recname,
                                             recname-REFER,
                                             recname-LIST,
                                             recname-AREA,
                                             ENDP.
                     3.  MOVE recname-REFER TO REFER.

FFM       SST        1.  MOVE recname-LINK TO recname-REFER.
                     2.  CALL "DATBAS" USING READV,
                                             STATUS,
                                             recname,
                                             recname-REFER,
                                             recname-LINKPATH,
                                             recname-LIST,
                                             recname-AREA,
                                                    ENDP.
                     3.  MOVE recname-REFER TO REFER.

FFR       RK         1.  MOVE ISQL-VAR-nn TO
                  CONTROL-recname.
                     2.  CALL "DATBAS" USING READM,
                                             STATUS,
                                             recname,
```

```
                                                CONTROL-recname,
                                                recname-LIST,
                                                recname-AREA,
                                                    NDP.
                    3.  MOVE SPACE TO REFER.

FNA       RAS       1.  CALL "DATBAS" USING RDNXT,
                                                STATUS,
                                                recname,
                                                recname-REFER,
                                                recname-LIST,
                                                recname-AREA,
                                                    ENDP.
                    2.  MOVE recname-REFER TO REFER.

FNM       SST       1.  CALL "DATBAS" USING READV,
                                                STATUS,
                                                recname,
                                                recname-REFER,
                                                recname-LINKPATH,
                                                recname-LIST,
                                                recname-AREA,
                                                    ENDP.
                    2.  MOVE recname-REFER TO REFER.

FNR       RK        1.  MOVE "END". TO REFER.

FOW       SST       1.  MOVE data-field TO
                        CONTROL-recname.
                    2.  CALL "DATBAS" USING READM,
                                                STATUS,
                                                recname,
                                                CONTROL-recname,
                                                recname-LIST,
                                                recname-AREA,
                                                    ENDP.
                    3.  MOVE SPACE TO REFER.

GF2         FG2             1.  MOVE "set-value" TO
                                ISQL-VAR-nn.
                                or
                                MOVE "set-value" TO
                                ISQR-VAR-nn.

GIF     RF1             1.  IF D-dfno = null-value
                            MOVE 1 TO RES-NULL-nn
                            MOVE ZEROES TO RES-nn
                        2.  IF D-dfno not = null-value
                            MOVE 0 TO RES-NULL-nn
                            MOVE D-dfno TO RES-nn
```

GIO	FG1	1. IF D-dfno = null-value MOVE 1 TO ISQL-NULL-nn MOVE ZEROES TO ISQL-VAR-nn 2. IF D-dfno NOT = null-value MOVE 0 TO ISQL-NULL-nn MOVE D-dfno TO ISQL-VAR-nn or 1. IF D-dfno = null value MOVE 1 TO ISQR-NULL-nn MOVE 0 TO ISQR-VAR-nn 2. IF D-dfno = null-value MOVE 0 TO ISQR-NULL-nn MOVE D-dfno TO ISQR-VAR-nn
GI1	-	NO-OP
GI2	-	NO-OP
IF	IT2	1. IF "set-value" OP ISQL-VAR-nn or IF "set-value" OP IS-VAR-nn
IFC	RS5	1. IF ((D-dfno NOT = null-value AND ISQL-NULL = 1) AND (D-dfno op ISQL-VAR-nn)) or OR ((D-dfno NOT = null-value AND ISQL-NULL NOT = 1) AND (D-dfno op ISQL-VAR-nn)) or IF ((D-dfno NOT = null-value AND ISQR-NULL NOT = 1) AND (D-dfno op ISQR-VAR-nn)) or OR ((D-dfno NOT = null-value AND ISQR-NULL NOT = 1) AND (D-dfno op ISQR-VAR-nn))
IFX	0C4	1. IF INDEX-dfno > INDEX-dfno-MAX
IF1	RK1	1. ADD 1 TO LOOP-COUNT 2. IF LOOP-COUNT > MAX GO TO EXIT-(n-1).
IF2	RK2	1. IF LOOP-COUNT = n
IIF	CIF	1. IF statement for WHERE clause qualification (internal schema)

IN1	RS1	1. IF NOT (D-dfno1 = null-value AND D-DFNO2 = null-value) AND NOT (D-dfno1 OP D-dfno2)
IN2	RS4	1. IF NOT (D-dfno = null-value AND ISQL-NULL-nn = 1) AND NOT (D-dfno op ISQL-VAR-nn) or IF NOT (D-dfno = null-value AND ISQR-NULL-nn = 1) AND NOT (D-dfno op ISQR-VAR-nn)
IN3	RS3	1. IF NOT ("set-value" op ISQL-VAR-nn) or IF NOT ("set-value" op ISQR-VAR-nn)
IRF	SST	1. IF (SUCCESSFUL AND NOT END-OR RECORDS)
IRN	SST	1. IF (END-OF-RECORDS AND SUCCESSFUL)
LOP	NN	1. LOOP-nn.
MR1	-	NO-OP
MR2	-	NO-OP
MV	FUS	1. MOVE MSG-VARI-nn TO D-dfno or MOVE null-value TO D-dfno
MVI	OC2	1. MOVE ISQL-VAR-nn TO INDEX-dfno-MAX.
MVK	RK2	1. MOVE ISQL-VAR-nn TO D-dfno
MVM	OC3	1. MOVE nn TO INDEX-dfno-MAX. or 2. MOVE D-dfno TO INDEX-dfno-MAX.
MVS	MVS	1. MOVE MSG-VAR-nn TO ISQL-VAR-nn.

MVX	OC5	1. MOVE D-dfno (INDEX-dfno,...) TO RES-nn.
MVY	OC6	1. MOVE null-value TO D-dfno (INDEX-dfno,...)
MVZ	-	1. MOVE ZERO TO LOOP-COUNT.
MV3	FG3	1. MOVE MSG-VAR-nn TO PARM-alg-modinst-parmno. 2. MOVE ZERO TO PARM-NULL-alg-modinst.
MV4	FG4	1. MOVE constant-value TO PARM-alg-modinst-parmno.
MV5	FU2	1. MOVE PARM-alg-modinst-parmno TO rtid-AREA 2. MOVE ZERO TO PARM-NULL-alg-modinst.
MV6	FU1	1. MOVE PARM-alg-modinst-parmno TO D-dfno 2. MOVE ZERO TO PARM-NULL-alg-modinst.
MV7	FU3	1. IF D-dfno = null-value MOVE 1 TO PARM-NULL-alg-modinst MOVE ZEROES TO PARM-alg-modinst-parmno 2. IF D-dfno NOT = null-value MOVE 0 TO PARM-NULL-alg-modinst MOVE D-dfno TO PARM-alg-modinst-parmno
MV8	FU4	1. MOVE rtid-AREA TO PARM-alg-modinst-parmno 2. MOVE 0 TO PARM-NULL-alg-modinst.
NLP	NN	1. GO TO EXIT-nn. (+1)
NXS	-	1. NEXT SENTENCE
OU1	-	NO-OP
OU2	RF2	1. MOVE "set-value" TO RES-nn.

```
OU3     RF3          1.  IF PARM-NULL-alg-modinst = 1
                         MOVE 1 TO RES-NULL-nn
                         MOVE ZEROES TO RES-nn
                     2.  IF PARM-NULL-alg-modinst = 0
                         MOVE 0 TO RES-NULL-nn
                         MOVE PARM-alg-modinst-parmno
                              TO RES-nn

OU4     RF3          1.  IF PARM-NULL-alg-modinst = 1
                         MOVE 1 TO TAG-NULL-tagno
                         MOVE ZEROES TO TAG-tagno
                     2.  IF PARM-NULL-alg-modinst = 0
                         MOVE 0 TO TAG-NULL-tagno
                         MOVE PARM-alg-modinst-parmno
                             TO TAG-tagno

OU5     OU5          1.  IF D-dfno = null-value
                         MOVE 1 TO TAG-NULL-tagno
                         MOVE ZEROES TO TAG-tagno
                     2.  IF D-dfno NOT = null-value
                         MOVE 0 TO TAG-NULL-tagno
                         MOVE D-dfno TO TAG-tagno

PIO     N/A          1.  WRITE RESULTS-REC.
                     2.  ADD 1 TO REC-COUNT.

RCR     N/A          1.  MOVE SPACE TO REFER.

RDK     RD1          1.  MOVE keyvalue TO CONTROL-recname.
                     2.  CALL "DATBAS" USING DEL-M,
                                             STATUS,
                                             recname,
                                             CONTROL-recname,
                                             recname-LIST,
                                             recname-AREA,
                                                     ENDP.
                     3.  MOVE SPACE TO REFER.

RDS       RD2        1.  CALL "DATBAS" USING DELVD,
                                             STATUS,
                                             recname,
                                             recname-REFER,
                                             recname-LINKPATH,
                                             CONTROL-recname,
                                             recname-LIST,
                                             recname-AREA,
                                                     ENDP.

RIK       RI1        1.  MOVE keyvalue TO CONTROL-recname.
                     2.  CALL "DATBAS" USING ADD-M,
                                             STATUS,
```

```
                                                    recname,
                                                    CONTROL-recname,
                                                    recname-LIST,
                                                    recname-AREA,
                                                           ENDP.

RIS         RI2         1.  CALL "DATBAS" USING ADDVC,
                                                    STATUS,
                                                    recname,
                                                    recname-REFER,
                                                    recname-LINKPATH,
                                                    CONTROL-recname,
                                                    recname-LIST,
                                                    recname-AREA,
                                                    ENDP.
                        2.  MOVE recname-REFER TO REFER.

RUK         RU1         1.  MOVE keyvalue TO CONTROL-recname.
                        2.  CALL "DATBAS" USING WRITM,
                                                    STATUS,
                                                    recname,
                                                    CONTROL-recname,
                                                    recname-LIST,
                                                    recname-AREA,
                                                           ENDP.
                        3.  MOVE SPACE TO REFER.

RUS         RU2         1.  CALL "DATBAS" USING WRITV,
                                                    STATUS,
                                                    recname,
                                                    recname-REFER,
                                                    recname-LINKPATH,
                                                    CONTROL-recname,
                                                    recname-LIST,
                                                    recname-AREA,
                                                           ENDP.
                        2.  MOVE recname-REFER TO REFER.

SXI         OC2         1.  SET INDEX-dfno TO ISQL-VAR-nn.

SX1         OC1         1.  SET INDEX-dfno TO 1.

UIF         UIF         1.  IF statement for WHERE clause
                            qualification for union
                            discriminators

XLP          NN         1.  GO TO EXIT-nn.
```

```
CIF     CIF     1. IF statement for WHERE clause
                   qualification (conceptual schema)

CAL     CAL     1.  IF PARM-NULL-alg-modinst = 1
                          GO TO alg-modinst
                2.  CALL "alg-id" USING
                    PARM-alg-modinst-parmno
                          .
                          .
                          .
                          .
                3.  error handling
                4.  alg-modinst

EIF     N/A     1.  . (period)

ELP     NN      1.  GO TO LOOP-nn.
                2.  EXIT-nn.

ELS     N/A     1.  ELSE

EP      -       1.  EXIT-nn. (+1)

FFA     RAS       1.  MOVE BEGIN TO recname-REFER.
                  2.  CALL "DATBAS" USING RDNXT,
                                          STATUS,
                                          recname,
                                          recname-REFER,
                                          recname-LIST,
                                          recname-AREA,
                                          ENDP.
                  3.  MOVE recname-REFER TO REFER.

FFM     SST       1.  MOVE recname-LINK TO recname-REFER.
                  2.  CALL "DATBAS" USING READV,
                                          STATUS,
                                          recname,
                                          recname-REFER,
                                          recname-LINKPATH,
```

```
                                        recname-LIST,
                                        recname-AREA,
                                                ENDP.
                    3.  MOVE recname-REFER TO REFER.

FFR         RK      1.  MOVE ISQL-VAR-nn TO
                        CONTROL-recname.
                    2.  CALL "DATBAS" USING READM,
                                        STATUS,
                                        recname,
                                        CONTROL-recname,
                                        recname-LIST,
                                        recname-AREA,
                                                NDP.
                    3.  MOVE SPACE TO REFER.

FNA         RAS     1.  CALL "DATBAS" USING RDNXT,
                                        STATUS,
                                        recname,
                                        recname-REFER,
                                        recname-LIST,
                                        recname-AREA,
                                                ENDP.
                    2.  MOVE recname-REFER TO REFER.

FNM         SST     1.  CALL "DATBAS" USING READV,
                                        STATUS,
                                        recname,
                                        recname-REFER,
                                        recname-LINKPATH,
                                        recname-LIST,
                                        recname-AREA,
                                                ENDP.
                    2.  MOVE recname-REFER TO REFER.

FNR         RK      1.  MOVE "END". TO REFER.

FOW         SST     1.  MOVE data-field TO
                        CONTROL-recname.
                    2.  CALL "DATBAS" USING READM,
                                        STATUS,
                                        recname,
                                        CONTROL-recname,
                                        recname-LIST,
```

recname-AREA,
ENDP.

3. MOVE SPACE TO REFER.

GF2	FG2	1. MOVE "set-value" TO ISQL-VAR-nn. or MOVE "set-value" TO ISQR-VAR-nn.
GIF	RF1	1. IF D-dfno = null-value MOVE 1 TO RES-NULL-nn MOVE ZEROES TO RES-nn 2. IF D-dfno not = null-value MOVE 0 TO RES-NULL-nn MOVE D-dfno TO RES-nn
GIO	FG1	1. IF D-dfno = null-value MOVE 1 TO ISQL-NULL-nn MOVE ZEROES TO ISQL-VAR-nn 2. IF D-dfno NOT = null-value MOVE 0 TO ISQL-NULL-nn MOVE D-dfno TO ISQL-VAR-nn or 1. IF D-dfno = null value MOVE 1 TO ISQR-NULL-nn MOVE 0 TO ISQR-VAR-nn 2. IF D-dfno = null-value MOVE 0 TO ISQR-NULL-nn MOVE D-dfno TO ISQR-VAR-nn
GI1	-	NO-OP
GI2	-	NO-OP
IF	IT2	1. IF "set-value" OP ISQL-VAR-nn or IF "set-value" OP IS-VAR-nn

```
IFC     RS5          1.  IF ((D-dfno NOT = null-value AND
                         ISQL-NULL = 1) AND
                         (D-dfno op ISQL-VAR-nn))
                         or
                         OR ((D-dfno NOT = null-value AND
                             ISQL-NULL NOT = 1) AND
                         (D-dfno op ISQL-VAR-nn))
                         or
                         IF ((D-dfno NOT = null-value AND
                           ISQR-NULL NOT = 1) AND
                         (D-dfno op ISQR-VAR-nn))
                         or
                         OR ((D-dfno NOT = null-value AND
                         ISQR-NULL NOT = 1) AND
                         (D-dfno op ISQR-VAR-nn))

IFX     0C4          1.  IF INDEX-dfno > INDEX-dfno-MAX

IF1     RK1          1.  ADD 1 TO LOOP-COUNT
                     2.  IF LOOP-COUNT > MAX
                                 GO TO EXIT-(n-1).

IF2     RK2          1.  IF LOOP-COUNT = n

IIF     CIF          1.  IF statement for WHERE clause
                        qualification (internal schema)

IN1     RS1          1.  IF NOT (D-dfno1 = null-value AND
                         D-DFNO2 = null-value)
                         AND NOT (D-dfno1 OP D-dfno2)

IN2     RS4          1.  IF NOT (D-dfno = null-value AND
                         ISQL-NULL-nn = 1)
                         AND NOT (D-dfno op ISQL-VAR-nn)
                                 or
                        IF NOT  (D-dfno = null-value AND
                         ISQR-NULL-nn = 1)
                         AND NOT (D-dfno op ISQR-VAR-nn)

IN3     RS3          1.  IF NOT ("set-value" op
                                 ISQL-VAR-nn)
                         or
```

		IF NOT ("set-value" op ISQR-VAR-nn)
IRF	SST	1. IF (SUCCESSFUL AND NOT END-OR RECORDS)
IRN	SST	1. IF (END-OF-RECORDS AND SUCCESSFUL)
LOP	NN	1. LOOP-nn.
MR1	-	NO-OP
MR2	-	NO-OP
MV	FUS	1. MOVE MSG-VARI-nn TO D-dfno or MOVE null-value TO D-dfno
MVI	OC2	1. MOVE ISQL-VAR-nn TO INDEX-dfno-MAX.
MVK	RK2	1. MOVE ISQL-VAR-nn TO D-dfno
MVM	OC3	1. MOVE nn TO INDEX-dfno-MAX. or 2. MOVE D-dfno TO INDEX-dfno-MAX.
MVS	MVS	1. MOVE M G-VAR-nn TO ISQL-VAR-nn.
MVX	OC5	1. MOVE D-dfno (INDEX-dfno,...) TO RES-nn.
MVY	OC6	1. MOVE null-value TO D-dfno (INDEX-dfno,...)
MVZ	-	1. MOVE ZERO TO LOOP-COUNT.

MV3	FG3	1. MOVE MSG-VAR-nn TO PARM-alg-modinst-parmno. 2. MOVE ZERO TO PARM-NULL-alg-modinst.
MV4	FG4	1. MOVE constant-value TO PARM-alg-modinst-parmno.
MV5	FU2	1. MOVE PARM-alg-modinst-parmno TO rtid-AREA 2. MOVE ZERO TO PARM-NULL-alg-modinst.
MV6	FU1	1. MOVE PARM-alg-modinst-parmno TO D-dfno 2. MOVE ZERO TO PARM-NULL-alg-modinst.
MV7	FU3	1. IF D-dfno = null-value MOVE 1 TO PARM-NULL-alg-modinst MOVE ZEROES TO PARM-alg-modinst-parmno 2. IF D-dfno NOT = null-value MOVE 0 TO PARM-NULL-alg-modinst MOVE D-dfno TO PARM-alg-modinst-parmno
MV8	FU4	1. MOVE rtid-AREA TO PARM-alg-modinst-parmno 2. MOVE 0 TO PARM-NULL-alg-modinst.
NLP	NN	1. GO TO EXIT-nn. (+1)
NXS	-	1. NEXT SENTENCE
OU1	-	NO-OP
OU2	RF2	1. MOVE "set-value" TO RES-nn.

```
OU3     RF3          1.  IF PARM-NULL-alg-modinst = 1
                         MOVE 1 TO RES-NULL-nn
                         MOVE ZEROES TO RES-nn
                     2.  IF PARM-NULL-alg-modinst = 0
                         MOVE 0 TO RES-NULL-nn
                         MOVE PARM-alg-modinst-parmno
                              TO RES-nn

OU4     RF3          1.  IF PARM-NULL-alg-modinst = 1
                         MOVE 1 TO TAG-NULL-tagno
                         MOVE ZEROES TO TAG-tagno
                     2.  IF PARM-NULL-alg-modinst = 0
                         MOVE 0 TO TAG-NULL-tagno
                         MOVE PARM-alg-modinst-parmno
                             TO TAG-tagno

OU5     OU5          1.  IF D-dfno = null-value
                         MOVE 1 TO TAG-NULL-tagno
                         MOVE ZEROES TO TAG-tagno
                     2.  IF D-dfno NOT = null-value
                         MOVE 0 TO TAG-NULL-tagno
                         MOVE D-dfno TO TAG-tagno

PIO     N/A          1.  WRITE RESULTS-REC.
                     2.  ADD 1 TO REC-COUNT.

RCR     N/A          1.  MOVE SPACE TO REFER.

RDK     RD1          1.  MOVE keyvalue TO CONTROL-recname.
                     2.  CALL "DATBAS" USING DEL-M,
                                             STATUS,
                                             recname,
                                             CONTROL-recname,
                                             recname-LIST,
                                             recname-AREA,
                                                     ENDP.
                     3.  MOVE SPACE TO REFER.

RDS       RD2        1.  CALL "DATBAS" USING DELVD,
                                             STATUS,
                                             recname,
                                             recname-REFER,
```

```
                                                recname-LINKPATH,
                                                CONTROL-recname,
                                                recname-LIST,
                                                recname-AREA,
                                                        ENDP.

RIK         RI1         1.  MOVE keyvalue TO CONTROL-recname.
                        2.  CALL "DATBAS" USING ADD-M,
                                                STATUS,
                                                recname,
                                                CONTROL-recname,
                                                recname-LIST,
                                                recname-AREA,
                                                        ENDP.

RIS         RI2         1.  CALL "DATBAS" USING ADDVC,
                                                STATUS,
                                                recname,
                                                recname-REFER,
                                                recname-LINKPATH,
                                                CONTROL-recname,
                                                recname-LIST,
                                                recname-AREA,
                                                ENDP.
                        2.  MOVE recname-REFER TO REFER.

RUK     RU1             1.  MOVE keyvalue TO CONTROL-recname.
                        2.  CALL "DATBAS" USING WRITM,
                                                STATUS,
                                                recname,
                                                CONTROL-recname,
                                                recname-LIST,
                                                recname-AREA,
                                                        ENDP.
                        3.  MOVE SPACE TO REFER.

RUS         RU2         1.  CALL "DATBAS" USING WRITV,
                                                STATUS,
                                                recname,
                                                recname-REFER,
                                                recname-LINKPATH,
                                                CONTROL-recname,
                                                recname-LIST,
                                                recname-AREA,
                                                        ENDP.
```

		2.	MOVE recname-REFER TO REFER.
SXI	OC2	1.	SET INDEX-dfno TO ISQL-VAR-nn.
SX1	OC1	1.	SET INDEX-dfno TO 1.
UIF	UIF	1.	IF statement for WHERE clause qualification for union discriminators
XLP	NN	1.	GO TO EXIT-nn.

TABLE 25-2

LIBRARY NAME - TOTAL

MACRO NAME - QPT01

PARAMETERS - p1 - Request Processor Program Name

```
IDENTIFICATION DIVISION.
PROGRAM-ID.     p1.
ENVIRONMENT DIVISION.
```

TABLE 25-2

LIBRARY NAME - TOTAL

MACRO NAME - QPT02

PARAMETERS - None

```
*
*  TOTAL KEYWORDS
*
 01  ADDVC                    PIC X(5) VALUE "ADDVC".
 01  ADD-M                    PIC X(5) VALUE "ADD-M".
 01  CLOSX                    PIC X(5) VALUE "CLOSX".
 01  COMIT                    PIC X(5) VALUE "COMIT".
 01  DELVD                    PIC X(5) VALUE "DELVD".
 01  DEL-M                    PIC X 5) VALUE "DEL-M".
 01  ENDP                     PIC X(4) VALUE "END.".
 01  OPENX                    PIC X(5) VALUE "OPEN".
 01  RDNXT                    PIC X(5) VALUE "RDNXT".
 01  READM                    PIC X(5) VALUE "READM".
 01  READV                    PIC X(5) VALUE "READV".
 01  SINON                    PIC X(5) VALUE "SINON".
 01  SINOF                    PIC X(5) VALUE "SINOF".
 01  WRITM                    PIC X(5) VALUE "WRITM".
 01  WRITV                    PIC X(5) VALUE "WRITV".
*
*  TOTAL STATUS VALUES
*
 01  STATUS                   PIC X(4).
     88  SUCCESSFUL                    VALUE "****".
     88  CONTROL-FIELD-BLANK           VALUE "BCTL".
     88  MASTER-NOT-FOUND              VALUE "MRNF".
     88  LINK-PATH-INVALID             VALUE "MLNF".
     88  FILE-ALREADY-OPEN             VALUE "DUPO".
     88  NO-SINOF-ISSUED               VALUE "EXSO".
 01  BEGIN                    PIC X(4) VALUE "BEGN".
 01  REFER                    PIC X(5).
     88 END-OF-RECORDS                 VALUE "END.".
 01  FILE-OPEN              PIC 9.
     88 FILE-IS-OPEN                      VALUE 1.
```

TABLE 25-2

TOTAL REQUEST PROCESSOR MACROS

LIBRARY NAME - TOTAL

MACRO NAME - QPT01

PARAMETERS - p1 - Request Processor Program Name

```
IDENTIFICATION DIVISION.
PROGRAM-ID.     p1.
ENVIRONMENT DIVISION.
```

LIBRARY NAME - TOTAL

MACRO NAME - QPT02

PARAMETERS - None

```
*
*  TOTAL KEYWORDS
*
 01  ADDVC                    PIC X(5) VALUE "ADDVC".
 01  ADD-M                    PIC X(5) VALUE "ADD-M".
 01  CLOSX                    PIC X(5) VALUE "CLOSX".
 01  COMIT                    PIC X(5) VALUE "COMIT".
 01  DELVD                    PIC X(5) VALUE "DELVD".
 01  DEL-M                    PIC X(5) VALUE "DEL-M".
 01  ENDP                     PIC X(4) VALUE "END.".
 01  OPENX                    PIC X(5) VALUE "OPEN".
 01  RDNXT                    PIC X(5) VALUE "RDNXT".
 01  READM                    PIC X(5) VALUE "READM".
 01  READV                    PIC X(5) VALUE "READV".
 01  SINON                    PIC X(5) VALUE "SINON".
 01  SINOF                    PIC X(5) VALUE "SINOF".
 01  WRITM                    PIC X(5) VALUE "WRITM".
 01  WRITV                    PIC X(5) VALUE "WRITV".
*
*  TOTAL STATUS VALUES
*
 01  STATUS                   PIC X(4).
     88  SUCCESSFUL                    VALUE "****".
     88  CONTROL-FIELD-BLANK           VALUE "BCTL".
     88  MASTER-NOT-FOUND              VALUE "MRNF".
     88  LINK-PATH-INVALID             VALUE "MLNF".
     88  FILE-ALREADY-OPEN             VALUE "DUPO".
     88  NO-SINOF-ISSUED               VALUE "EXSO".
 01  BEGIN                    PIC X(4) VALUE "BEGN".
 01  REFER                    PIC X(5).
     88 END-OF-RECORDS                 VALUE "END.".
 01  FILE-OPEN              PIC 9.
     88 FILE-IS-OPEN                      VALUE 1.
```

LIBRARY NAME - TOTAL

MACRO NAME - QPT03

PARAMETERS - qqqqq - Request Processor Program Name

```
01  MESG-DESC                  PIC X(60).
01  MODULE-NAME                PIC X(10) VALUE "qqqqq".
01  RET-STATUS                 PIC X(5).
01  MY-HOST                    PIC XXX VALUE SPACE.
COPY ERRCDM OF IISSCLIB.
COPY ERRFS OF IISSCLIB.
01  FCB1                       PIC S9(9) COMP.
01  RECORD-LENGTH                   PIC S9(9) COMP.
01  ACCESS-MODE               PIC X.
01  NUMBER-OF-RECORDS          PIC S9(9) COMP VALUE 2000.
01  ERROR-FLAG                 PIC X(5).
```

LIBRARY NAME - TOTAL

MACRO NAME - QPT04

PARAMETERS - None

```
01  MESSAGE-BODY-OUT.
    03  OUTFILE-NAME          PIC X(30).
    03  REC-COUNT             PIC 9(6)    VALUE ZERO.
    03  QP-STATUS             PIC X(5).
01  GLOBAL-REALM.
    03  FILLER                PIC X(6).
    03  FILLER                PIC X(13) OCCURS 40 TIMES.
    03  FILLER                PIC X(4).
01  REALM-FILE-COUNT          PIC 99.
```

LIBRARY NAME - TOTAL

MACRO NAME - QPT05

PARAMETERS - p1 - DBMOD Name
p2 - Request Processor Program Name

```
*
*  TOTAL SINON/SINOF, OPENX/CLOSX
*  WORKING STORAGE VARIABLES
*
 01  TOTAL-ACCESS              PIC X(6)  VALUE "RDONLY".
 01  DBMOD                     PIC X(8)  VALUE "p1".
 01  TASK                      PIC X(8)  VALUE "p2".
 01  OPTIONS                   PIC X(14) VALUE "LOGOPTS=N,END.".
```

LIBRARY NAME - TOTAL

MACRO NAME - QPT06

PARAMETERS - None

```
PROCEDURE DIVISION USING MESSAGE-BODY-IN
                         GLOBAL-REALM
                         MESSAGE-BODY-OUT
START-HERE.
      MOVE KES-SUCCESSFUL  TO QP-STATUS.
      MOVE ZERO            TO REC-COUNT.
      MOVE ZERO            TO FILE-OPEN.
      CALL "TOTOPN" USING REALM-OPEN
                          GLOBAL-REALM
                          TOTAL-STATUS.
      IF SUCCESSFUL
         NEXT SENTENCE
     ELSE
         MOVE KES-TOTAL-OPEN-FAILED TO RET-STATUS
         STRING "TOTAL OPENX FAILED WITH STATUS-"
            DELIMITED BY SIZE
         TOTAL-STATUS DELIMITED BY SIZE
            INTO MESG-DESC
         PERFORM TOTAL-ABORT THRU PGM-END.
```

LIBRARY NAME - TOTAL

MACRO NAME - QPT08

PARAMETERS - None

```
   CALL "DATBAS" USING CLOSX,
                       STATUS,
                       REALM-CLOSE,
                       ENDP.
   IF SUCCESSFUL
      NEXT SENTENCE
   ELSE
      MOVE KES-TOTAL-CLOSX-FAILED TO RET-STATUS
      STRING "TOTAL CLOSX FAILED WITH STATUS-"
         DELIMITED BY SIZE
         STATUS DELIMITED BY SIZE INTO MESG-DESC
      PERFORM TOTAL-ABORT THRU PGM-END.
   CALL "DATBAS" USING SINOF,
                       STATUS,
                       TASK,
                       ENDP.
   IF SUCCESSFUL
      NEXT SENTENCE
   ELSE
      MOVE KES-TOTAL-SINOF-FAILED TO RET-STATUS
      STRING "TOTAL SINOF FAILED WITH STATUS-"
         DELIMITED BY SIZE
         STATUS DELIMITED BY SIZE INTO MESG-DESC
      PERFORM TOTAL-ABORT THRU PGM-END.
```

LIBRARY NAME - TOTAL

MACRO NAME - QPT09

PARAMETERS - None

```
     MOVE "00000" TO QP-STATUS.
     GO TO 999-EOJ.
  REPORT-ERROR.
     MOVE "ABORT" TO QP-STATUS.
 *
 * Request processors will use the ERRPRO mailbox
 * logging facility to report all errors.
 *
      PERFORM PROCESS-ERROR.
  REPORT-ERROR-EXIT.
  PGM-ABORT.
  TOTAL-ABORT.
      PERFORM PROCESS-ERROR.
      IF FILE-IS-OPEN
      CLOSE RESULTS.
  999-EOJ.
     EXIT PROGRAM.
  PGM-END.
```

LIBRARY NAME - TOTAL

MACRO NAME - READM

PARAMETERS - recname - record name to be retrieved

```
   CALL "DATBAS" USING READM,
                       STATUS,
                       recname,
                       CONTROL-recname,
                       recname-LIST,
                       recname-AREA,
                       ENDP.
   MOVE SPACE TO REFER.
   IF SUCCESSFUL OR MASTER-NOT-FOUND
      NEXT SENTENCE
   ELSE
      MOVE KES-TOTAL-READM-FAILED TO RET-STATUS
      STRING "TOTAL READM FAILED WITH STATUS OF"
                   DELIMITED BY SIZE
             STATUS DELIMITED BY SIZE
             "FOR RECORD" DELIMITED BY SIZE
             recname DELIMITED BY SIZE
      INTO MESG-DESC
      PERFORM REPORT-ERROR THRU REPORT-ERROR-EXIT.
```

LIBRARY NAME - TOTAL

MACRO NAME - READV

PARAMETERS - recname - record name to be retrieved

```
   CALL "DATBAS"  USING READV,
                        STATUS,
                        recname,
                        recname-REFER,
                        recname-LINKPATH,
                        recname-LIST,
                        recname-AREA,
                        ENDP.
   MOVE recname-REFER TO REFER.
   IF SUCCESSFUL
      NEXT SENTENCE
   ELSE
      MOVE KES-TOTAL-READV-FAILED TO RET-STATUS
      STRING "TOTAL READV FAILED WITH STATUS OF"
                   DELIMITED BY SIZE
             STATUS DELIMITED BY SIZE
             "FOR RECORD" DELIMITED BY SIZE
             recname DELIMITED BY SIZE
      INTO MESG-DESC
      PERFORM REPORT-ERROR THRU REPORT-ERROR-EXIT.
```

LIBRARY NAME - TOTAL

MACRO NAME - RDNXT

PARAMETERS - recname - record name to be retrieved

```
   CALL "DATBAS" USING RDNXT,
                       STATUS,
                       recname,
                       recname-REFER,
                       recname-LIST,
                       recname-AREA,
                       ENDP.
   MOVE recname-REFER TO REFER.
   IF SUCCESSFUL
      NEXT SENTENCE
   ELSE
      MOVE KES-TOTAL-RDNXT-FAILED TO RET-STATUS
      STRING "TOTAL RDNXT FAILED WITH STATUS OF"
                    DELIMITED BY SIZE
             STATUS DELIMITED BY SIZE
             "FOR RECORD" DELIMITED BY SIZE
             recname DELIMITED BY SIZE
      INTO MESG-DESC
      PERFORM REPORT-ERROR THRU REPORT-ERROR-EXIT.
```

LIBRARY NAME - TOTAL

MACRO NAME - PI0

PARAMETERS - nn - data retrieval looping costruct

EXIT-nn.

```
LIBRARY NAME - TOTAL

MACRO NAME   - ADDM

PARAMETERS   - recname


 CALL "DATBAS" USING ADD-M,
                     STATUS,
                     recname,
                     CONTROL-recname,
                     recname-LIST,
                     recname-AREA,
                     ENDP.
 MOVE SPACE TO REFER.
 IF SUCCESSFUL
    NEXT SENTENCE
 ELSE
   MOVE KES-TOTAL-ADD-M-FAILED TO RET-STATUS
 STRING "TOTAL ADD-M FAILED WITH STATUS OF" DELIMITED BY SIZE

     STATUS
DELIMITED BY SIZE
       "FOR RECORD"                          DELIMITED BY SIZE
          recname                              DELIMITED BY SIZE
      INTO MESG-DESC
      PERFORM REPORT-ERROR THRU REPORT-ERROR-EXIT
```

```
LIBRARY NAME - TOTAL

MACRO NAME   - DELM

PARAMETERS   - recname

 CALL "DATBAS" USING DEL-M,
                     STATUS,
                     recname,
                     CONTROL-recname,
                     recname-LIST,
                     recname-AREA,
                     ENDP.
 MOVE SPACE TO REFER.
  IF SUCCESSFUL
    NEXT SENTENCE
 ELSE
    MOVE KES-TOTAL-DEL-M-FAILED TO RET-STATUS
    STRING "TOTAL DEL-M FAILED WITH STATUS OF" DELIMITED BY SIZE
    STATUS                                     DELIMITED BY SIZE
    "FOR RECORD"                               DELIMITED BY SIZE
    recname                                    DELIMITED BY SIZE
     INTO MESG-DESC
     PERFORM REPORT-ERROR THRU REPORT-ERROR-EXIT.
```

LIBRARY NAME - TOTAL

MACRO NAME - WRITM

PARAMETERS - recname

```
   CALL "DATBAS" USING WRITM,
                       STATUS,
                       recname,
                       CONTROL-recname,
                       recname-LIST,
                       recname-AREA,
                       ENDP.
   MOVE SPACE TO REFER.
   IF SUCCESSFUL
      NEXT SENTENCE
   ELSE
      MOVE KES-TOTAL-WRITM-FAILED TO RET-STATUS
       STRING "TOTAL DEL-M FAILED WITH STATUS OF" DELIMITED BY
SIZE
       STATUS                                    DELIMITED BY SIZE
       "FOR RECORD"                              DELIMITED BY SIZE
        recname                                   DELIMITED BY SIZE
          INTO MESG-DESC
          PERFORM REPORT-ERROR THRU REPORT-ERROR-EXIT.
```

LIBRARY NAME - TOTAL

MACRO NAME - ADDVC

PARAMETERS - recname

```
   CALL "DATBAS" USING ADDVC,
                       STATUS,
                       recname,
                       recname-REFER,
                       recname-LINKPATH,
                       CONTROL-recname,
                       recname-LIST,
                       recname-AREA,
                       ENDP.
   MOVE recname-REFER TO REFER.
   IF SUCCESSFUL
      NEXT SENTENCE
   ELSE
    MOVE KES-TOTAL-ADDVC-FAILED TO RET-STATUS
     STRING "TOTAL ADDVC FAILED WITH STATUS OF" DELIMITED BY

     SIZE
     STATUS                                  DELIMITED BY SIZE
     "FOR RECORD"                            DELIMITED BY SIZE
     recname                                 DELIMITED BY SIZE
          INTO MESG-DESC
          PERFORM REPORT-ERROR THRU REPORT-ERROR-EXIT.
```

LIBRARY NAME - TOTAL

MACRO NAME - DELVD

PARAMETERS - recname

```
   CALL "DATBAS" USING DELVD,
                       STATUS,
                       recname,
                       recname-REFER,
                       recname-LINKPATH,
                       CONTROL-recname,
                       recname-LIST,
                       recname-AREA,
                       ENDP.
   MOVE recname-REFER TO REFER.
   IF SUCCESSFUL
      NEXT SENTENCE
   ELSE
    MOVE KES-TOTAL-DELVD-FAILED TO RET-STATUS
     STRING "TOTAL DELVD FAILED WITH STATUS OF" DELIMITED BY

SIZE
     STATUS                                  DELIMITED BY SIZE
     "FOR RECORD"                            DELIMITED BY SIZE
      recname                                DELIMITED BY SIZE
      INTO MESG-DESC
      PERFORM REPORT-ERROR THRU REPORT-ERROR-EXIT.
```

LIBRARY NAME - TOTAL

MACRO NAME - WRITV

PARAMETERS - recname

```
   CALL "DATBAS" USING WRITV,
                       STATUS,
                       recname,
                       recname-REFER,
                       recname-LINKPATH,
                       CONTROL-recname,
                       recname-LIST,
                       recname-AREA,
                       ENDP.
   MOVE recname-REFER TO REFER.
   IF SUCCESSFUL
      NEXT SENTENCE
   ELSE
      MOVE TOTAL-WRITV-FAILED TO RET-STATUS
          STRING "TOTAL WRITV FAILED WITH STATUS OF" DELIMITED

BY SIZE
      STATUS                                    DELIMITED BY SIZE
      "FOR RECORD"                              DELIMITED BY SIZE
       recname                                   DELIMITED BY SIZE
      INTO MESG-DESC
          PERFORM REPORT-ERROR THRU REPORT-ERROR-EXIT.
```

LIBRARY NAME - IBM

MACRO NAME - IBM01

PARAMETERS - None

```
01  R062A10.
    02  DYNAME       PIC X(8)     VALUE "R062A10".
01  PLIST.
    02  DDNAME       PIC X(8)     VALUE "SYS010".
    02  DSNAME       PIC X(44)    VALUE "  ".
    02  DSMEMBER     PIC X(8)     VALUE "  ".
    02  DSPSWD       PIC X(8)     VALUE "  ".
    02  DSSTATUS     PIC X(8)     VALUE "  ".
    02  DSNDISP      PIC X(8)     VALUE "  ".
    02  DSADISP      PIC X(8)     VALUE "  ".
    02  DSUNIT       PIC X(8)     VALUE "  ".
    02  RSVD1        PIC X(8)     VALUE "  ".
    02  DSVOLSER     PIC X(6)     VALUE "  ".
    02  RSVD2        PIC X(40)    VALUE "  ".
    02  RSVOLREF     PIC X(44)    VALUE "  ".
    02  DSFREE       PIC X(8)     VALUE "  ".
    02  DSLABEL      PIC X(4)     VALUE "  ".
    02  DSINOUT      PIC X(4)     VALUE "  ".
    02  RSVD3        PIC X(16)    VALUE "  ".
    02  DSPWDLBL     PIC X(8)     VALUE "  ".
    02  DSDATE       PIC X(02)    VALUE "  ".
    02  DSALLOC      PIC X(5)     VALUE "  ".
    02  DSPRI        PIC X(6)     VALUE "  ".
    02  DSSEC        PIC X(6)     VALUE "  ".
    02  DSDIR        PIC X(6)     VALUE "  ".
    02  DSRLSE       PIC X(8)     VALUE "  ".
    02  DSCONTIG     PIC X(8)     VALUE "  ".
    02  DSROUND      PIC X(8)     VALUE "  ".
    02  RSVD4        PIC X(24)    VALUE "  ".
    02  DSBLKSI      PIC X(5)     VALUE "  ".
    02  DSORG        PIC X(8)     VALUE "  ".
    02  DSKEYLEN     PIC X(3)     VALUE "  ".
    02  DSLRECL      PIC X(5)     VALUE "  ".
    02  DSRECFM      PIC X(8)     VALUE "  ".
    02  DSDCBDS      PIC X(44)    VALUE "  ".
    02  RSVD5        PIC X(24)    VALUE "  ".

01  ERRBLK.
    02  ERRCODE      PIC X(2).
    02  ERRINFO      PIC X(2).
    02  ALLOCRC      PIC X(4).
```

LIBRARY NAME - IBM

MACRO NAME - IBM02

PARAMETERS - None

```
*
* Free DD name
*
    MOVE SPACES TO DSSTATUS.
    MOVE SPACES TO DSNDISP.
    MOVE SPACES TO DSADISP.
    MOVE LOW-VALUE TO ERRBLK.
    CALL R062A10 USING PLIST, ERRBLK.
    IF ALLOCRC EQUAL ZERO
        NEXT SENTENCE
    ELSE
        PERFORM PGM-ABORT THRU 999-EOJ.
*
* Allocate DD name to RESULTS file
*
    MOVE RESFILE TO DSNAME.
    MOVE `SHR' TO DSSTATUS.
    MOVE `KEEP' TO DSNDISP.
    MOVE `KEEP' TO DSADISP.
    CALL R062A10 USING PLIST, ERRBLK.
    IF ALLOCRC EQUAL ZERO
        NEXT SENTENCE
    ELSE
        PERFORM PGM-ABORT THRU 999-EOJ.
*
* Open file.
*
    OPEN OUTPUT RESULTS.
```

SECTION 26

FUNCTION PRE9.5 GENERATE IMS REQUEST PROCESSOR

This function is a compile-time module (CDQPI) whose purpose is to generate the COBOL source code necessary to satisfy an NDML subtransaction request against an IMS database. The IMS RP extracts data along a hierarchical access path using IMS as the DBMS and IBM as the host computer, and then reformats the data into a sequential file. Two reference tables are contained in this section.

26.1 Inputs

Inputs are the outputs from function PRE5 CS to IS Decomposer.

1. The following data is retrieved from the NTM Input Message:

   Database Identification Code
   Subtransaction Identification Number
   Request Processor Program Name
   IMS PSB Name
   Source/Work File Names
   Error File Name for User's AP
   Number of Entries in Tables
   Packed Tables

2. IS NDML Subtransactions, represented by the IS-ACTION-LIST and the IS-QUALIFY-LIST.

3. Conceptual schema formats.

   The Result Field Table contains the CS descriptions of the data fields requested by the NDML transaction.

   The CS-QUALIFY-LIST contains the CS description of the variables and constants which are passed to the IMS RP at run-time.

4. Owner/Member relationships

   The Set Table contains information about the sets that must be traversed in order to process the transaction.

5. CDM Requirements

   The CDM entity classes that must be accessed to generate an IMS Request Processor are:

   IMS implementation of RECORD_TYPE:
        IMS_SEGMENT_SIZE (81)
   Data field/IMS segment mapping:
        DATA_FIELD (79) SEGMENT_DATA_FIELD (82)
   PSB/PCB mapping: PSB_PCB (78)

The CDM attribute classes that must be populated by the generator are:

MODID of GENERATED_AP_PSB (10)
PSB_NAME of GENERATED_AP_PSB (10)

26.2 Internal Requirements

1. The IMS-ACCESS-PATH table contains a list of segment names in hierarchical order, which represent the IMS retrieval path required to satisfy the NDML subtransaction.

```
01  IMS-ACCESS-PATH.
    03  IAP-MAX                PIC 99   VALUE 15.
    03  IAP-USED               PIC 99.
    03  IAP-ENTRY  OCCURS 15 TIMES INDEXED BY
                        IAP-INDEX.
        05  IAP-SEGNAME            PIC X(8).
        05  IAP-SEG-BYTES          PIC 9(5).
        05  IAP-ACTION             PIC X.
        05  IAP-QUALIFY            PIC X.
```

2. The DATA-DEFINITION-TABLE contains a list of the data fields and their descriptions for each of the segments in the IMS-ACCESS-PATH.

```
01  DATA-DEFINITION-TABLE.
    03  DD-SEGMENT  OCCURS 15 TIMES INDEXED BY
                        IAP-INDEX.
        05  DD-MAX                 PIC 99 VALUE 25.
        05  DD-FIELDS-USED         PIC 99.
        05  DD-FIELD  OCCURS 25 TIMES INDEXED BY
                            DD-INDEX.
            07  DD-DFID                PIC X(30).
            07  DD-IMS-IND             PIC X.
            07  DD-START-BYTE          PIC 9(5).
            07  DD-TYPE                PIC X.
            07  DD-SIZE                PIC 9(3).
            07  DD-ND                  PIC 99.
            07  DD-IS-PTR              PIC 99.
            07  DD-ISQ-PTR             PIC 99.
            07  DD-ISQ-TYPE            PIC X.
```

3. An extension to the IS-QUALIFY-LIST tables, represented by the ISQ-DD-POINTERS table. The pointer table links the retrieval and qualifying fields on the right side of type 3 subtransactions to their position within an IMS segment.

```
01  ISQ-DD-POINTERS.
    03  FILLER OCCURS 25 TIMES INDEXED BY
                    ISQ-INDEX.
        05  ISQ-SEG-PTR            PIC 99.
        05  ISQ-FLD-PTR            PIC 99.
```

26.3 Constraints

1. The current release does not support a confluent hierarchy where one 1:m and one or more 1:1 relationships exist at the same level in the hierarchy.

2. From the CDM point of view, each PCB corresponds to a single database.

3. Queries on fields not explicitly defined to IMS are supported. The field name value of any field specified on an NDML transaction must be defined to the CDM as the DFID value on a DATA_FIELD (67) entity class occurrence. If the field name also appears on a FIELD statement in a DBD gen, the value "K" must be assigned to IMS_DF_IND on the SEGMENT_DATA_FIELD (82) occurrence. If the field name does not appear in a DBD gen, the value "U" must be assigned to IMS_DF_IND.

4. Secondary data structures are supported in so far as they are set up properly with a PCB. IMS secondary indexes will be queried by the request processor provided the XDFLD name specified in the DBD gen is specified as the DFID value on a DATA FIELD (67) occurrence.

5. Hexadecimal data types are not supported.

6. Access to GSAM files will not be allowed.

7. Each IMS Request Processor will be a batch oriented BMP to allow concurrent use of data base files. A request processor will not make use of any IMS message capabilities nor will it contain IMS checkpoint calls. It should be noted that a BMP will degrade the performance of the on-line IMS system if the transactions it processes are not qualified on at least one key field.

26.4 Outputs

IMS Request Processor program source code.

1. The Data Division of each IMS RP consists of:

   a) FILE SECTION containing the file description of the output RESULTS-REC.

   b) WORKING-STORAGE section containing the data definitions of IMS function codes, IMS status codes, segment search arguments, segment I/O area, NTM message, NTM return codes, accept status and system status, and parameter list.

   c) LINKAGE SECTION containing the PCB mask definitions.

2. The procedure division of each IMS RP consists of:

   a) Entry call to DL1 to establish the link between the program and the IMS data base.

   b) If the RP is remote, calls to the NTM to initialize network communications and receive any run-time search values, or the values are input as parameters.

   c) Call to dynamically allocate the output file if the RP is on a host that is remote.

   d) Conceptual to internal data transform on run-time search parameters.

   e) Processing loop containing:

      1. Call to DLI to retrieve one or more segments from the IMS database. The call contains four elements:

         - Function code: The function is always 'get unique' the first time the loop is executed and 'get next' thereafter.

         - PCB mask: The PCB mask definition is obtained from CDM meta data. IMS returns information about the retrieval call in the PCB mask.

         - I/O receiving area: The working storage area for the segments being retrieved. A segment consists of one or more data fields and is the smallest chunk of data that can be accessed. The number of segments to be retrieved and their definitions are derived from the various input tables and CDM meta data.

         - Segment search arguments: A segment search argument is constructed for each segment to be retrieved. If a segment contains a data field which is also a qualifying field, the qualification test is embedded in its segment search argument.

      2. Qualification checks for non-IMS fields. An NDML transaction can search on data fields contained in an IMS database but not explicitly defined to IMS. When this occurs, COBOL IF tests are generated for the non-IMS fields. An implied AND exists between each IF test.

      3. Internal to conceptual data transform on retrieved data.

      4. Write to the output file.

f) Status code checks and error processing routines.

g) If the RP is on a remote host, call to NTM with completion status message.

h) Standard request processor error handling procedures.

26.5 Processing

Initialize the IMS Request Processor generator (CDQPI). If the RP and the calling AP reside on the same host, then the tables containing data will be passed as parameters. If not, then the RP will receive its message through the "INITIAL" and "RCV" calls of the NTM. All the tables in the input message are packed. Unpack the tables into the formats described in the Inputs section using the number-of-entries variables.

The input tables contain entries for all the subtransactions which will be used to process the original NDML statement. However, only one subtransaction is processed at a time. For any given input table, the generator will examine only those entries whose subtrans-id is equal to the input message subtrans-id (QPGI-SUBTRANS-ID).

The IMS Request Processor generator utilizes two sequential work files, to allow working-storage and procedure division statements to be constructed simultaneously. Work File 1 contains all the data division statements and Work File 2 contains all the procedure division statements. At the end of processing the CDCWF routine is called to combine the two work files and place resulting source code on Work File 1.

Code which is common to all IMS Request Processors is generated utilizing the macro replacement utility CDMACR. Code specific to an IMS Request Processor is derived from information in the various tables, the NTM input message and CDM meta data.

Errors are reported using one of three error routines, depending on the type of error. User errors are reported by 'RPTERR', system errors by 'ERRPRO'. User errors consist of CDM meta data errors. All other types of errors, such as table overflows, DBMS errors, NTM errors, etc., are considered system errors. Note that NTM errors are reported twice, using ERRPRO and SIGERR. The generator may encounter both user and system errors. A request processor can encounter only system errors.

1. Establish IMS Access Path

   Count and process all entries in the SET TABLE where ST-SUBTRANS-ID is equal to QPGI-SUBTRANS-ID.

1.1 Locate the top of the access path.

Compare ST-OWNER (ST-INDEX) to the first occurrence of ST-MEMBER in all other SET TABLE entries. NOTE: For an IMS database a set has one and only one member.

If a match is found, increment ST-INDEX and continue at 1.1. If a match is not found, the top of the access path has been located.

Set IAP-INDEX to 1.

Set IAP-SEGNAME (IAP-INDEX) to ST-OWNER (ST-INDEX).

1.2 Add segment to IMS Access Path table.

Increment IAP-INDEX.

Set IAP-SEGNAME (IAP-INDEX) to ST-MEMBER (ST-INDEX, 1).

Reset ST-INDEX to 1.

1.3 Find next segment in access path. Compare IAP-SEGNAME (IAP-INDEX) to ST-OWNER (ST-INDEX).

If a match is found, continue at 1.2.

If a match is not found:

If ST-INDEX is less than ST-USED, increment ST-INDEX and continue at 1.3. Otherwise:

If IAP-INDEX equal the number of entries in the SET TABLE subtrans id group +1, continue at 1.4.

Generate a BAD-ACCESS-PATH user error message and terminate processing.

1.4 Finish Access Path Table Construction

Set IAP-USED to IAP-INDEX.

For each IAP-ENTRY perform 1.4.1 through 1.4.4.

1.4.1 Look up segment size.

Access IMS_SEGMENT_SIZE (81) using DB_ID and RT_ID, where DB_ID equals subtrans DBID and RT_ID equals IAP-SEGNAME (IAP-INDEX).

Set IAP-SEG-BYTES (IAP-INDEX) to SEGMENT-SIZE.

1.4.2 Flag segments for data retrieval.

Compare IAP-SEGNAME to each RT_ID in the IS-ACTION-LIST where IS-SUBTRANS-ID equals QPGI-SUBTRANS-ID.

If a match is found, set IAP-ACTION (IAP-INDEX) to 'R'.

If a match is not found, set IAP-ACTION (IAP-INDEX) to space.

1.4.3 Flag segments for data qualifications.

Set IAP-QUALIFY (IAP-INDEX) to space.

Compare IAP-SEGNAME (IAP-INDEX) to each entry in the IS-QUALIFY-LIST where ISQ-TYPE is equal to '2' or '3' and ISQ-SUBTRANS-IDL is equal to QPGI-SUBTRANS-ID.

IF ISQ-TYPE is equal to '2'

Compare IAP-SEGNAME (IAP-INDEX) to ISQ-RTIDL (ISQ-INDEX). If ISQ-RTIDL is blank, then compare IAP-SEGNAME to ISQ-RTIDR.

If equal, set IAP-QUALIFY (IAP-INDEX) to 'Q'.

If ISQ-TYPE is equal to '3'

If ISQ-RTIDL (ISQ-INDEX) is equal to ISQ-RTIDR (ISQ-INDEX), continue at 1.4.4.

Compare IAP-SEGNAME (IAP-INDEX) to ISQ-RTIDL (ISQ-INDEX). If equal, set IAP-QUALIFY (IAP-INDEX) to 'Q' and continue at 1.4.4.

Compare IAP-SEGNAME (IAP-INDEX) to ISQ-RTIDR (ISQ-INDEX). If equal, set IAP-QUALIFY (IAP-INDEX) to 'Q'.

1.4.4 Delete unnecessary segment entries.

If both IAP-ACTION and IAP-QUALIFY are space, move spaces to IAP-SEGNAME.

1.5 Check for empty access path table.

It is possible for all IAP-SEGNAME entries to be blank. If this occurs, generate a NO-DATA user error message and terminate processing.

2. Complete Access Path

2.1 For each new table name (IS-RTID) found in the retrieve list perform the following:

2.2 SELECT Data Field entity (E67) using DB name and Record name as key.

2.3 If the entry has the DBMS-ACCESSABLE flag set then store the following in the SEG-TABLE.

```
DD-TFID  =  FILE NAME
DD-DFID  =  FIELD NAME
DD-TYPE  =  TYPE_ID
DD-SIZE  =  SIZE_ID
DD-ND    =  MAX_DF_DEC_LEN
```

2.4 Set CURRENT-GROUP-ID = blank

2.5 SELECT Component Data Field (E195) using DB-OBJ-NO, RT-OBJ-NAME, DF-OBJ-NAME as key (i.e., Am I a "GROUP LEADER").

2.5.1 If a row is returned, then set

CURRENT-GROUP-ID = FIELD-OBJECT-NAME

2.5.2 If no row is found then

SELECT Component Data field (E195) using DB-OBJ-NO, RT-OBJ-NAME, DF-OBJ-NAME, DF-SEQ-NO as key (i.e., Am I a member of a group)

If a row is returned, then set CURRENT GROUP ID = GROUP DF OBJ NAME

2.6 CHECK the STACK for GROUP-ID.

2.6.1 Check the CURRENT-GROUP-ID with the STACK-GROUP-ID

2.6.2 If not equal POP the STACK and go to 2.6.1

2.7 Generate a level number from the current STACK-LEVEL-NO

2.8 Generate the DF-OBJECT-NAME

2.9 If this field is a group identifier, do the following:

2.9.1 ADD 2 to the current level number

2.9.2 PUSH new level number and new GROUP-ID onto the top of the stack

2.10 SELECT a Data Field Redefinition entity (E196) using DB-OBJ-NO, RT-OBJ-NAME, DF-OBJ-NAME as key.

2.11 If an entry is found, generate a REDEFINES clause

2.12 If the Number of Data Fields field of (E67) is non-zero, then generate an OCCURS clause.

2.13 SELECT an Elementary Data Field entity (E194)

2.14 If an entry is found, then generate a PICTURE clause

2.15 If not last entry, go to 5.

2.16 SELECT the next Data Field Entity (E67) from the CDM1 database. If the entry has the DBMS-ACCESSIBLE flag set, then store the values as in 2.3.

3. Generate IMS Request Processor Program

3.1 Generate IDENTIFICATION DIVISION, ENVIRONMENT DIVISION, and DATA DIVISION/FILE SECTION.

Call "CDMACR" utility with library name equal to "IMS", macro name equal to "QPG01" replacing "qqqqq" with the AP name passed by the NTM (QPGI-QP-NAME).

3.2 Generate the conceptual data definition for retrieved data in the RESULTS-REC file I/O area and working storage.

Call "CDRFT" using work-file-1, subtrans-id and the RESULT-FIELD-TABLE.

3.3 Generate data statements for IMS function codes and status codes.

Call "CDMACR" with library name equal to "IMS" and macro name equal to "QPI01".

3.4 Generate internal data definitions for runtime search parameters.

Call "CDPRM" using work-file-1, QPGI-SUBTRANS-ID, and IS-QUALIFY-LIST.

3.5 Generate Segment Search Arguments (SSAs)

Generate an SSA definition for each entry in the IMS Access Path Table where IAP-SEGNAME is not blank. SSAs must be generated in the same order as the entries in the IMS Access Path Table. Set IAP-INDEX to 1.

Establish a count for IMS qualifying fields and a count for non-IMS qualifying fields. Set each count to zero.

3.5.1 Generate unqualified portion of SSA.

Call "CDMACR" with library name equal to "IMS", macro-name equal to "QPI02" replacing:

qqq by IAP-SEGNAME value
n by IAP-INDEX value

If current IAP-QUALIFY equals space, increment IAP-INDEX, set IMS qualifying field count to zero and continue at 3.5.1.

Generate search parameter field definitions.

For each DD-FIELD entry in the DATA-DEFINITION-TABLE for the current segment:

3.5.1.1 If DD-ISQ-PTR equals zero or DD-ISQ-TYPE not equal to '2' or DD-IMS-IND equal to 'U', continue at 3.5.1.2.

Generate definition of qualifying field within SSA.

Add 1 to IMS qualifying field count. If count greater than 1, generate:

```
03  FILLER      PIC X VALUE "&".
```

Call "CDMACR" using "IMS" as the library name and "QPI03" as the macro name replacing:

'dfid' with DD-DFID value
'op' with ISQ-OP (DD-ISQ-PTR) value
'size' with DD-SIZE value
'qq' with DD-ISQ-PTR

Note: The relational operator value contained in ISQ-OP may require conversion to IMS notation. Refer to Figure 9-5 for conversion values.

3.5.1.2 Increment DD-INDEX.

ISQ Rel Op	IMS Rel OP	COBOL Rel Op	Description
=	=	=	equal to
!=	NE	NOT=	not equal to
>	>	>	greater than
<	<	<	less than
>=	>=	NOT<	greater than or equal to
<=	<=	NOT>	less than or equal to

Figure 26-2. Relational Operators

3.6 Generate Segment I/O Area

Generate:

01 SEGMENT-AREA.

For each DD-FIELD entry in the SEG-TABLE Table, generate an 03 level data definition statement.

03 dfid-xx picture clause.

Replace "dfid" with DD-DFID. Replace "xx" with DD-IS-PTR. Call "CDPIC" to generate the picture clause using the work file name, DD-TYPE, DD-SIZE and DD-ND.

3.7 Generate data definitions required for parameter interface.

3.7.1 If the RP is on a remote host:

Generate NTM run-time parameters.

Call "CDMACR" using "IMS" as the library name and "QPGNTM" as the macro name.

Generate format of message sent to NTM.

Call "CDMACR" using "IMS" as the library name and "QPG03" as the macro name.

3.8 Generate conceptual schema format definitions for the variables and constants which will be input by the user at runtime.

Call "CDMSG" using work-file-1, QPGI-SUBTRANS-ID, CS-QUALIFY-LIST and IS-QUALIFY-LIST.

3.9 Generate the internal data definitions for retrieved data fields. These definitions will be used by the internal to conceptual conversion routine.

Call "CDRDF" using work-file-1, QPGI-SUBTRANS-ID, and IS-QUALIFY-LIST.

3.10 Generate data definitions required for dynamically allocating the RESULTS file on an IBM system.

Call "CDMACR" using "IBM" as the library name and "IBM01" as the macro name.

4. Generate LINKAGE SECTION.

4.1 Generate a PCB mask for each PCB associated with the current PSB.

Access PSB_PCB (78) in PCB_SEQ_NO order using PSB_NAME where PSB_NAME equals the PSB name from NTM input message.

For each occurrence of PSB_PCB (78):

Call "CDMACR" using "IMS" as the library name and "QPI07" as the macro name. Replace "q" with PCB_SEQ_NO and "size" with KEY_FEEDBACK_LEN.

Compare current dbid to DBID of PSB_PCB (78). If equal, then save the PCB_SEQ_NO for use in the "CBLTDLI" call in step 5.1.

4.2 Update CDM with generated program id.

Access GENERATED_AP_PSB (10) using PSB_NAME where PSB_NAME equals QPGI-PSBNAME. If it does not exist, create a new occurrence; otherwise, update GENERATED_MOD_ID using QPGI-QP-NAME.

5. Generate PROCEDURE DIVISION

5.1 Generate IMS entry call. This must be the first executable statement.

```
PROCEDURE DIVISION.
START-HERE.
     ENTRY 'DLITCBL' USING PCB list.
```

where list is the list of PCB_SEQ_NO values retrieved in step 4.

5.2 Generate code to initialize NTM and receive input message.

Call "CDMACR" using "IMS" as the library name and "QPG04X" as the macro name.

5.3 Generate code to dynamically free DD name, to allocate it to the RESULTS file, and to open the RESULTS file.

Call "CDMACR" using "IBM" as the library name and "IBM02" as the macro name.

5.4 Generate code to convert runtime search parameters from conceptual to internal format.

For each DD-FIELD entry in the SEG-TABLE:

Call "CDCI" using work-file-1, work-file-2, next parameter no., DD-ISQ-PTR, current dbid, DD-DFID, IAP-SEGNAME and TAG-NUMBER. All parameters must be 01 level parameters. TAG-NUMBER is the CSQ-AUCL from the CS-QUALIFY-LIST (CSQ-AUCR if CSQ-AUCL is blank) which is pointed to by ISQ-CSQ-PTR (DD-ISQ-PTR).

Generate code to move the internal format of the runtime search parameters to the proper segment search arguments.

If DD-IMS-IND is equal to 'K' generate:

```
MOVE ISQ-VAR-nn to SSA-ISQ-VAR-nn.

where nn = DD-ISQ-PTR.
```

5.5 Generate IMS data retrieval loop.

Generate:

```
     MOVE GU TO FUNC.
FETCH-LOOP.
     MOVE SPACES TO SEGMENT-AREA.
     CALL 'CBLTDLI' USING FUNC
                          PCB-n
                          SEGMENT AREA
                          SSA list.
     PERFORM STATUS-CODE-CHECK THRU STATUS-CHECK-EXIT.
```

where:

n	=	PCB-SEQ-NO saved in step 4
SSA list	=	list of SSAs generated in step 3.5.

5.6 Generate COBOL statements for non-IMS field qualification checks and for field-to-field qualification checks.

For each DD-FIELD entry in the DATA-DEFINITION-TABLE where DD-ISQ-TYPE is equal to '2' or '3':

If DD-ISQ-TYPE is equal to '2' and DD-IMS-IND is equal to 'U', generate the following statements:

```
IF ISQ-VAR-nn rel-op dfid-xx
   NEXT SENTENCE
ELSE
   GO TO FETCH-LOOP.
```

where:

dfid	=	DD-DFID
rel-op	=	COBOL version of ISQ-OP (DD-ISQ-PTR). Refer to Figure 9-5 for conversion values.
nn	=	DD-ISQ-PTR
xx	=	DD-IS-PTR (may be zero)

If DD-ISQ-TYPE is equal to '3', generate the following statements:

```
IF dfid-nn rel-op dfidr-xx
   NEXT SENTENCE
ELSE GO TO FETCH-LOOP
```

where:

dfid	=	current DD-DFID
nn	=	current DD-IS-PTR (may be zero)
rel-op	=	COBOL version of ISQ-OP (DD-ISQ-PTR). Refer to Figure 9-5 for conversion values.
dfidr	=	comparison DD-DFID whose IAP-INDEX = ISQ-SEG-PTR value pointed to by current DD-ISQ-PTR
DD-INDEX	=	ISQ-FLD-PTR value pointed to by current DD-ISQ-PTR
xx	=	comparison DD-IS-PTR

5.7 Transform retrieved data from internal to conceptual format and move to output RESULTS-REC.

For each DD-FIELD entry in the DATA-DEFINITION-TABLE where DD-IS-PTR is non-zero:

Generate a COBOL statement to move data from the segment area to the working storage area expected by the IS/CS transform routine.

MOVE dfid-nn to IS-VAR-nn.

where:

dfid = DD-DFID
nn = DD-IS-PTR

Call "CDIC" using work-file-2, next parameter no., DD-IS-PTR, current dbid, DD-DFID, IAP-SEGNAME and TAG-NUMBER. All parameters must be 01 level parameters. TAG-NUMBER is the RFT-ATTR pointed to by IS-RFT-PTR (DD-IS-PTR).

5.8 Generate remainder of Procedure Division

Call "CDMACR" using "IMS" as the library name and "QPG04" as the macro name. Replace 'q' with PCB_SEQ_NO saved in Step 4.

6. When the entire RP has been generated, return to PRE13 to continue generating code to satisfy the NDML request.

TABLE 26-1

IMS REQUEST PROCESSOR MACROs

LIBRARY: IBM
MACRO: IBM01

```
01  R062A10.
    02  DYNAME      PIC X(8)     VALUE 'R062A10'.
01  PLIST.
    02  DDNAME      PIC X(8)     VALUE 'SYS010'.
    02  DSNAME      PIC X(44)    VALUE '  '.
    02  DSMEMBER    PIC X(8)     VALUE '  '.
    02  DSPSWD      PIC X(8)     VALUE '  '.
    02  DSSTATUS    PIC X(8)     VALUE '  '.
    02  DSNDISP     PIC X(8)     VALUE '  '.
    02  DSADISP     PIC X(8)     VALUE '  '.
    02  DSUNIT      PIC X(8)     VALUE '  '.
    02  RSVD1       PIC X(8)     VALUE '  '.
    02  DSVOLSER    PIC X(6)     VALUE '  '.
    02  RSVD2       PIC X(40)    VALUE '  '.
    02  RSVOLREF    PIC X(44)    VALUE '  '.
    02  DSFREE      PIC X(8)     VALUE '  '.
    02  DSLABEL     PIC X(4)     VALUE '  '.
    02  DSINOUT     PIC X(4)     VALUE '  '.
    02  RSVD3       PIC X(16)    VALUE '  '.
    02  DSPWDLBL    PIC X(8)     VALUE '  '.
    02  DSDATE      PIC X(02)    VALUE '  '.
    02  DSALLOC     PIC X(5)     VALUE '  '.
    02  DSPRI       PIC X(6)     VALUE '  '.
    02  DSSEC       PIC X(6)     VALUE '  '.
    02  DSDIR       PIC X(6)     VALUE '  '.
    02  DSRLSE      PIC X(8)     VALUE '  '.
    02  DSCONTIG    PIC X(8)     VALUE '  '.
    02  DSROUND     PIC X(8)     VALUE '  '.
    02  RSVD4       PIC X(24)    VALUE '  '.
    02  DSBLKSI     PIC X(5)     VALUE '  '.
    02  DSORG       PIC X(8)     VALUE '  '.
    02  DSKEYLEN    PIC X(3)     VALUE '  '.
    02  DSLRECL     PIC X(5)     VALUE '  '.
    02  DSRECFM     PIC X(8)     VALUE '  '.
    02  DSDCBDS     PIC X(44)    VALUE '  '.
    02  RSVD5       PIC X(24)    VALUE '  '.
```

TABLE 26-1 (Continued)

IMS REQUEST PROCESSOR MACROs

```
01  ERRBLK.
    02  ERRCODE      PIC X(2).
    02  ERRINFO      PIC X(2).
    02  ALLOCRC      PIC X(4).
```

TABLE 26-1 (Continued)

IMS REQUEST PROCESSOR MACROs

LIBRARY: IBM
MACRO: IBM02

```
      *
      * Free DD name
          MOVE SPACES TO DSSTATUS.
          MOVE SPACES TO DSNDISP.
          MOVE SPACES TO DSADISP.
          MOVE LOW-VALUE TO ERRBLK.
          CALL R062A10 USING PLIST, ERRBLK.
          IF ALLOCRC EQUAL ZERO
              NEXT SENTENCE
          ELSE
              PERFORM PGM-ABORT THRU 999-EOJ.
      *
      * Allocate DD name to RESULTS file
          MOVE RESFILE TO DSNAME.
          MOVE 'SHR' TO DSSTATUS.
          MOVE 'KEEP' TO DSNDISP.
          MOVE 'KEEP' TO DSADISP.
          CALL R062A10 USING PLIST, ERRBLK.
          IF ALLOCRC EQUAL ZERO
              NEXT SENTENCE
          ELSE
              PERFORM PGM-ABORT THRU 999-EOJ.
      *
      * Open file.
          OPEN OUTPUT RESULTS.
```

TABLE 26-1 (Continued)

IMS REQUEST PROCESSOR MACROs

LIBRARY: IMS
MACRO: QPG01

```
IDENTIFICATION DIVISION
PROGRAM-ID  qqqqq.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT RESULTS     ASSIGN TO RESFILE.
DATA DIVISION.
FILE SECTION.
FD  RESULTS
    LABEL RECORDS ARE STANDARD
```

TABLE 26-1 (Continued)

IMS REQUEST PROCESSOR MACROs

LIBRARY: IMS
MACRO: QPG03

```
01  MESSAGE-BODY-OUT.
    03  OUTFILE-NAME          PIC X(30).
    03  REC-COUNT             PIC 9(6)      VALUE ZERO.
    03  QP-STATUS             PIC X(5).

01  MSG-OUT-L                 PIC 9(4)      VALUE 41.
```

TABLE 26-1 (Continued)

IMS REQUEST PROCESSOR MACROs

LIBRARY: IMS
MACRO: QPGNTM

```
 01  BUFFER                  PIC X(7680).
 01  BUFFER-SIZE             PIC 9(6)      VALUE 7680.
 01  DATA-TYPE               PIC X         VALUE 'N'.
 01  NTM-DESTINATION         PIC X(5)
 01  LOGICAL-CHANNEL         PIC X(3).
 01  MESSAGE-TYPE            PIC XX        VALUE '01'.
 01  MESSAGE-SERIAL-NUMBER   PIC X(7).
 01  TERMINATION-STATUS      PIC X         VALUE SPACE.
 01  TIMEOUT-VALUE           PIC X(23)     VALUE SPACES.
 01  WAIT-FLAG               PIC 9         VALUE 1.
*
* NTM RETURN-CODES, ACCEPT-STATUS AND SYSTEM-STATES:
*
 COPY SRVRET OF IISSCLIB.
```

TABLE 26-1 (Continued)

IMS REQUEST PROCESSOR MACROs

LIBRARY: IMS
MACRO: QPI01

```
      *
      *Function codes (GU, GN, GNP) and status codes
      *apply to any IMS RP.
      *
       01  IMS-FUNCTION-CODES.
           03  FUNC                    PIC X(4).
           03  GU                      PIC X(4)      VALUE 'GU '.
           03  GN                      PIC X(4)      VALUE 'GN '.
           03  GNP                     PIC X(4)      VALUE 'GNP'.
       01  IMS-RETURN-CODE             PIC XX.
           88  GOOD                                  VALUE '  '.
           88  NOT-FOUND                             VALUE 'GE '.
           88  END-OF-DATA                           VALUE 'GB '.
           88  END-OF-SEG-TYP                        VALUE 'GK '.
           88  END-OF-PATH                           VALUE 'GA '.

      *
      *Miscellaneous program variables
      *

       01  STATUS-CODE                 PIC 9.
           88  SUCCESSFUL                 VALUE 0.
           88  UNSUCCESSFUL               VALUE 1.
           88  DONE                       VALUE 9.
       01  RET-STATUS                  PIC X(5).
       01  MESG-DESC                   PIC X(60).

      *
      *Need a copy statement to bring in list of error
      *codes. Expecting IMS-RETRIEVE-ERROR.
      *
```

TABLE 26-1 (Continued)

IMS REQUEST PROCESSOR MACROs

LIBRARY: IMS
MACRO: QPI02

```
*
*Segment search argument
*
 01  SSA-n.
     03  FILLER                  PIC X(8) VALUE 'qqq'.
     03  FILLER                  PIC XX   VALUE ' D'.
```

TABLE 26-1 (Continued)

IMS REQUEST PROCESSOR MACROs

LIBRARY: IMS
MACRO: QPI03

```
03  FILLER                 PIC X    VALUE '('.
03  FILLER                 PIC X(8) VALUE 'dfid'.
03  FILLER                 PIC XX   VALUE 'op'.
03  SSA-ISQ-VAR-qq         PIC X(size).
03  FILLER                 PIC X    VALUE ')'.
```

TABLE 26-1 (Continued)

IMS REQUEST PROCESSOR MACROs

LIBRARY: IMS
MACRO: QPI07

```
01  PCB-q.
    03  PCBq-DBDNAME          PIC X(8).
    03  PCBq-SEGLEVEL         PIC XX.
    03  PCBq-STATUS           PIC XX.
    03  PCBq-PROCOPT          PIC X(4).
    03  FILLER                PIC X(4).
    03  PCBq-SEGNAME          PIC X(8).
    03  PCBq-KFBLEN           PIC S9(5)  COMP.
    03  PCBq-SENSEG           PIC S9(5)  COMP.
    03  PCBq-KEY              PIC X(size).
```

TABLE 26-1 (Continued)

IMS REQUEST PROCESSOR MACROs

LIBRARY: IMS
MACRO: QPG04X

```
CALL 'INITAL' USING BUFFER,
                    BUFFER-SIZE,
                    SYSTEM-STATE,
                    RET-CODE.

IF INITAL-SUCCESSFUL
    NEXT SENTENCE
ELSE
    PERFORM PGM-ABORT THRU 999-EOJ.
CALL 'RCV' USING LOGICAL-CHANNEL,
                 WAIT-FLAG,
                 NTM-SOURCE,
                 MESSAGE-TYPE,
                 DATA-LENGTH,
                 MESSAGE-BODY-IN,
                 ACCEPT-STATUS,
                 MESSAGE-SERIAL-NUMBER.
IF RCV-NORMAL-MESSAGE
    NEXT SENTENCE
ELSE
    PERFORM PGM-ABORT THRU 999-EOJ.
```

TABLE 26-1 (Continued)

IMS REQUEST PROCESSOR MACROs

LIBRARY: IMS
MACRO: QPI04

```
           WRITE RESULTS-REC.
           ADD 1 TO REC-COUNT.
           GO TO FETCH-LOOP.
       END-FETCH.
           CLOSE RESULTS.
           MOVE ZERO TO QP-STATUS.
           GO TO 999-EOJ.
       STATUS-CODE-CHECK.
           IF GOOD
               MOVE ZERO TO STATUS-CODE
               GO TO STATUS-CHECK-EXIT
           ELSE
               IF NOT-FOUND
               OR END-OF-DATA
               OR END-OF-SEG-TYPE
               OR END-OF-PATH
                   MOVE 9 TO STATUS-CODE
                   GO TO STATUS-CHECK-EXIT.
           MOVE 1 TO STATUS-CODE.
           MOVE IMS-RETRIEVE-ERROR TO RET-STATUS.
           STRING 'IMS ', FUNC, ' ERROR  '
           'STATUS CODE ', PCB-STATUS-q
           ' DBD ', PCBq-DBDNAME
           ' SEGMENT ', PCBq-SEGNAME
           DELIMITED BY SIZE INTO MESG-DESC.
           PERFORM PROCESS-ERROR.
       STATUS-CHECK-EXIT.
           EXIT.
       PGM-ABORT.
           MOVE 'ABORT' TO QP-STATUS.
           MOVE '1' TO TERMINATION-STATUS.
       999-EOJ.
           MOVE RESFILE TO OUTFILE-NAME.
           MOVE NTM-SOURCE TO NTM-DESTINATION.
           MOVE SPACES TO TIMEOUT-VALUE.
           MOVE 'N' TO DATA-TYPE.
```

TABLE 26-1 (Continued)

IMS REQUEST PROCESSOR MACROs

```
*
*reply to distributed request supervisor with completion
*message:
*
     CALL 'NSEND' USING NTM-DESTINATION,
                        LOGICAL-CHANNEL,
                        TIMEOUT-VALUE,
                        DATA-TYPE,
                        MESSAGE-TYPE,
                        MSG-OUT-L,
                        MESSAGE-BODY,OUT,
                        ACCEPT-STATUS.
     CALL 'TRMNAT' USING TERMINATION-STATUS.


*
*TRMNAT does a COBOL STOP RUN.
*
 COPY ERRPRO OF IISSCLIB.
*
*    TRMNAT does a COBOL STOP RUN.
```

TABLE 26-2

CDM META DATA ERROR MESSAGES

COBOL Name	Message
BAD-ACCESS-PATH	dbid CDM META DATA ERROR - IMS ACCESS PATH NOT NAVIGABLE
NO-DATA	dbid EXTRANEOUS ACCESS PATH - ACCESS PATH IGNORED
NO-IMS-DATA	dbid SEGMENT_DATA_FIELD MISSING FOR dfid
BAD-DF-LENGTH	dbid OVERLAPPING DATA FIELDS in iap-segname
BAD-SEG-LENGTH	dbid SUM OF DATA FIELD LENGTHS WITHIN iap-segname NOT EQUAL TO SEGMENT FILE SIZE IN CDM

☆U.S. Government Printing Office:1992—648-127/62422